4 PLC

4.1 Uvod

Industrijski kontroler ili **PLC** od *Programmable Logic Controller* - programabilni logički kontroler.

PLC - digitalni elektronski uređaj koji poseduje programabilnu memoriju za smeštanje instrukcija kojima se realizuju specifične funkcije, kao što su logičke i aritmetičke operacije, redosledno izvršenje različitih akcija, odmeravanje vremenskih intervala, prebrojavanje događaja itd, a sve u cilju upravljanja različitim mašinama i procesima putem digitalnih i/ili analognih ulazno/izlaznih jedinica.

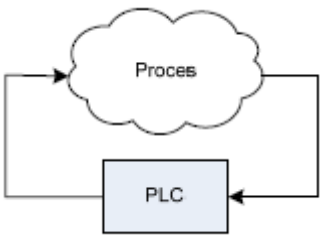
PLC - namenski elektronski **uređaj**, zasnovan na mikroprocesoru, koji je u mogućnosti da obavlja brojne tipove upravljačkih funkcija različitog nivoa složenosti.

PLC - **industrijski računar** čiji su hardver i softver posebno prilagođeni radu u industrijskim uslovima, a koji se lako može programirati i ugrađivati u nove i postojeće industrijske sisteme.

PLC (*Programmable*) - označava mogućnost programiranja. Program rada se priprema unapred i smešta u permanentu memoriju PLC kontrolera. PLC program se razvija u jeziku lestvičastih (***ladder***) dijagrama, koji je nastao po ugledu na tzv. *relejne šeme*.

P**L**C (*Logic*) - mogućnost obavljanja logičkih (Bulovih) funkcija je jedna od glavnih funkcionalnih karakteristika PLC kontrolera. PLC generiše diskretne (digitalne) izlazne signale u funkciji (logičkoj) diskretnih ulaznih signala - karakteristično za prvobitne tipove PLC kontrolera. Savremeni PLC-ovi, pored logičkih mogu dodatno da obavljaju aritmetičke operacije, odmeravaju vremenske intervale, prebrojavaju događaje, a prihvataju i generišu, pored diskretnih, i analogne signale.

PL**C** (*Controller*) - glavnu primenu PLC-ovi nalaze u industriji (proizvodnoj) gde se koriste za automatsko upravljanje procesima - prati ključne parametre procesa (posredstvom priključenih senzora i davača, i shodno memorisanom programu, generiše pobudu kojom deluje na proces (posredstvom aktuatora), Sl. 4-1.



Sl. 4-1 PLC - upravljačka jedinica procesa.

PLC vs. PC. PLC kontroler se razlikuje od računarskog sistema opšte namene po tome što ne poseduje spoljnu memoriju (diskove), kao i niz standardne ulazno/izlazne opreme (miš,

tastatura, ...). Pored toga, operativni sistem PLC-a je jednostavniji i pruža komparativno manje mogućnosti od računara opšte namene. Zapravo, PLC je koncipiran i projektovan za jedan relativno uzak i jasno definisan obim poslova vezanih za nadzor i upravljanje pojedinim uređajima, mašinama i procesima, što je rezultovalo u njegovoj izuzetnoj efikasnosti i jednostavnosti.

4.2. Istorijat PLC-ova

Nastanak PLC-ova se vezuje za kasne 60' i rane 70' godine prošlog veka, a nastali su na bazi konvencionalnih računara tog vremena. Njihova prvobitna primena bila je u automobilskoj industriji, a sa ciljem da se skrati vreme zastoja u proizvodnji usled promene proizvodnog procesa. Priprema pogona za proizvodnju novog modela automobila trajala je mesecima, što je zahtevalo prepovezivanje panela i ormara prepunih provodnika, relea, tajmera, sklopki i drugih, uglavnom elektro-mehaničkih komponenti, pomoću kojih su se, u to vreme, realizovale upravljačke funkcije. Uvođenje PLC-a, omogućilo je da se reprogramiranje upravljačke logike obavi ″preko tastature″ i da se, uz minimalna dodatna prepovezivanja, vreme zastoja proizvodnje skrati do tek nekoliko dana.

Glavni problem sa računarima/PLC-ovima iz 70' godina odnosio se upravno na njihovo reprogramiranje. Programi su bili pisani na niskom nivou (asembler), a time i komplikovani, a njih su mogli da sastavljaju samo visoko-stručni i iskusni programeri. Kasnih 70' godina učinjen je izvestan napredak u pogledu pojednostavljenja procedure programiranja. 1972. godine pojavljuju se mikroprocesori, koji, zahvaljujući povoljnom odnosu performanse/cena brzo nalaze široku primenu u mnogim oblastima elektronike, pa i industrijske automatizacije. Od tog vremena, pa do danas, PLC-ovi se neprestano usavršavaju, kako u pogledu performansi (moći obrade), softverske podrške, mogućnosti sprezanja sa najrazličitijim uređajima, tako u pogledu načina i procedura programiranja, kao i jezika koji se koriste za njihovo programiranje. Na taj način, PLC-ovi su postali ″razumljiviji″ široj populaciji industrijskih inženjera.

80' godina dolazi do eksponencijalnog rasta tržišta PLC-ova, a njihova primena se širi van industrijskih pogona, i to na sisteme kao što su elektro-energetska postrojenja, sistemi automatizacije stambenih/poslovnih objekata, sistemi obezbeđenja i nadzora itd. Danas, PLC-ovi se sve više primenjuju i u oblastima kao što je medicinska oprema i uređaji, kućni aparati i sl.

Prvi PLC kontroleri su bili jednostavni uređaji za **on/off** upravljanje i prevashodno su se koristili za zamenu zastarele relejne tehnike. Međutim, takvi PLC kontroleri nisu mogli da obezbede složenije upravljanje, kao što je upravljanje temperaturom, pritiskom, pozicijom. U međuvremenu, proizvođači PLC kontrolera razvili su i ugradili u PLC kontrolere brojna poboljšanja i funkcionalna unapređenja. Savremeni PLC kontroleri imaju mogućnost obavljanje izuzetno složenih zadataka kao što je upravljanje preciznim pozicioniranjem i upravljanje složenim tehnološkim procesima. Takođe, brzina rada PLC kontrolera je značajno povećana, kao i lakoća programiranja. Razvijeni su brojni moduli specijalne

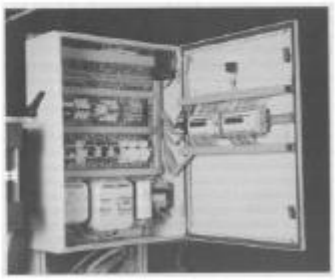
namene za primene kao što je radio komunikacija, vizija ili čak prepoznavanje govornih komandi.

Smatra se da inženjer sa znanjem relejne logike može ovladati osnovnim PLC funkcijama u roku od nekoliko sati. Slično važi i za inženjere sa znanjem digitalne elektronike. Za ovladavanje složenijim funkijama i mogućnostima PLC-a dovoljno je nekoliko nedelja ili jedan kurs.

Danas postoji veliki broj različitih tipova PLC kontrolera koji se razlikuju po veličini, izgledu i moći obrade, počev od malih jedinica sa malim i ograničenim brojem ulaza i izlaza do velikih, modularnih sistema koji se mogu konfigurisati za rad sa više stotina ili čak hiljada ulaza/izlaza. Na Sl. 4-2 je prikazan izgled PLC kontrolera iz familije ***Allen Bradley SLC 500 Modular Controllers*.** PLC se sastoji iz *šasije (rack)* koja ima određeni broj *slotova* u koje 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.

Sl. 4-2 Izgled komponenti modularnog PLC sistema.

Na Sl. 4-3(a) je prikazan kontrolni panel starijeg datuma koji se koristio u nekom proizvodnom pogonu za upravljanje procesom proizvodnje. Panel je velikih dimenzija i sadrži veliki broj žica, konektora, relae, što može stvoriti velike probleme u održavanju i servisiranju. Na Sl. 4-3(b) je prikazan PLC sistem koji u potpunosti zamenjuje relejni sistem sa Sl. 4-3(a). PLC sistem je manjih dimenzija i pouzdaniji. U slučaju promene načina rada, relejni sistem se mora prepovezati, dok se PLC može veoma brzo reprogramirati.



(a) (b)

Sl. 4-3 (a) Relejni panel; (b) PLC sistem

4.3 Prednosti i nedostaci PLC-ova

4.3.1 Prednosti

Fleksibilnost. U prošlosti, svaka električno-upravljana mašina za proizvodnju zahtevala je svoju sopstvenu upravljačku jedinicu; u pogonu sa 15 mašina, postojalo je 15 različitih, namenski projektovanih, upravljačkih jedinica. Danas je moguće isti model PLC-a koristiti za upravljanje bilo kojom od 15 mašina. Uz to, verovatno neće biti potrebe za 15 PLC-ova, jer jedan PLC lako može da opsluži više nezavisnih mašina, tako što će, konkurentno, za svaku priključenu mašinu izvršavati poseban, namenski program.

Laka promena programa i korekcija grešaka. Kod tradicionalnih, relejnih panela, svaka promena programa rada zahtevala je značajan utrošak vremena radi prepovezivanje panela i uređaja. S druge strane, kod PLC-a, promena programa je laka i brza. Novi program se preko tastature ili na neki drugi način učitava u PLC, a prepovezivanje obično nije potrebno, tako da celokupna aktivnost ne traje duže od nekoliko minuta. Takođe, uočene nepravilnosti u radu sistema, koje su posledica greške u programu mogu se lako i brzo ispraviti.

Veliki broj kontakta. Funkcija relejne upravljačke jedinice se realizuje povezivanjem relea (kontakta) koji imaju ulogu elektro-mehaničkih logičkih kola ili flip-flopova. Broj relea u jednom relejnom panelu je ograničen (npr. fizičkim dimenzijama panela), što ograničava i složenost funkcije koja se može realizovati. Može se desiti da u slučaju promene ″programa″ rada upavljačke jedice, broj raspoloživih relea više nije dovoljan da bi se realizovala nova funkcija. U takvim slučajevima, neophodno je ugraditi dodatne relee ili relejne blokove, što može biti dugotrajna operacija skopčana s neizbežnim mehaničkim intervencijama. S druge strane, PLC može realizovati funkcije čija je složenost ograničena jedino raspoloživom memorijom.

Niska cena. Uporedo s napretkom tehnologije, implementacione mogućnosti PLC-ova neprestano rastu. Danas je moguće, po ceni ispod 100$, nabaviti PLC sa ogromnim brojem interno-raspoloživih ″virtuelnih″ relea, tajmera, brojača, sekvencera i drugih funkcija.

Mogućnost probnog rada. Rad PLC-a se može ispitati u laboratoriji, pre ugradnje u proizvodni pogon. Program se piše, testira, analizira i, ako je neophodno, modifikuje sve do trenutka kada se proceni da su sve zahtevane funkcije korektno realizovane. Tek tada se program prenosi u PLC koji se instalira (ili je već instaliran) u proizvodni pogon. Na ovaj način, postiže se velika ušteda skupog „fabričkog" vremena (nema zastoja u proizvodnji). Nasuprot tome, testiranje konvencionalnih relejnih sistema se može obaviti samo u fabričkoj hali, što može biti veoma vremenski neracionalno.

Mogućnost vizuelnog praćenja rada. Rad PLC-a se može direktno pratiti na ekranu monitora - na pogodan način se u grafičkom obliku prikazuju stanja ulaza i izlaza PLC-a uz „osvetljeno" prikazivanje logičkih putanja koje su trenutno aktivne i ispisivanje obaveštenja o eventualnom neispravnom radu sistema ili o nastanku nekih izuzetnih situacija. Na taj način, svaka kritična situacija se može uočiti istog trenutka kada se i desila, što olakšava pronalaženje nastalih kvarova. Naime, kod naprednih PLC sistema, postoji mogućnost programiranja poruka za svako moguće neispravno ponašanje, koje će se prikazati onog trenutka kada se takva situacija detektuje od strane PLC-a, (kao npr. „MOTOR #7 je preopterećen").

Brzina rada. U poređenju sa releima koji su, s obzirom na svoju elektro-mehaničku prirodu, spori, brzina izvršenja PLC programa je veoma velika. Brzina rada PLC-a određena je trajanjem tzv. *sken ciklusa*, što je reda milisekundi. Drugim rečima, vreme koje protekne od trenutka kada se promeni stanje ulaza PLC-a do trenutka kada PLC-a reaguje postavljajući svoje izlaze tipično nije duže od nekoliko do maksimalno nekoliko desetina milisekundi.

Leder programiranje. Za programiranje PLC kontrolera koristi se jezik *lestvičastih logičkih dijagrama* (ili leder dijagrama - *ladder diagram*), koji je već dugi niz godina u upotrebi u industriji pri projektovanju logičkih i sekvencijalnih relejnih upravljačkih jedinica. Ovaj jezik koristi grafičku notaciju koja je po vizuelnom izgledu i logici rada slična dijagramima relejnih šema i zbog toga je lako razumljiv industrijskim inženjerima. Drugim rečima, industrijski inženjeri ne moraju biti eksperti za programiranje da bi u svojim sistemima koristili PLC-ove.

Pouzdanost i lakoća održavanja. Poluprovodničke komponente, od kojih je PLC sačinjen, su, generalno, pouzdanije od mehaničkih sistema ili relea i tajmera. Uz to, za realizaciju PLC-ovi se koriste komponente sa veoma visokim faktorom pouzdanosti. Iz tog razloga, troškovi održavanja upravljačkih sistema zasnovanih na PLC-u su niži, a vreme zastoja kraće.

Jednostavnost naručivanja komponenti upravljačkog sistema. PLC je jedan uređaj. Kada naručeni PLC stigne u industrijski pogon, svi brojači, relei, i druge ″virtuelne″ komponente ″sadržane″ u PLC-u su takođe stigle. Situacija je potpuno drugačija kada se projektuje relejni panel. Na primer, potrebno je 20 različitih tipova relea i tajmera koji se nabavljaju od 12 različitih dobavljača, sa različitim rokovima isporuke i potencijalnim problemima sa dostupnošću traženih komponenti. Ako projektant propusti da naruči samo jednu komponentu, celokupan projekat se odlaže za vreme isporuke te komponente. Kod PLC-ova

uvek postoji „jedan rele više“ - pod uslovom da je naručen PLC sa izvesnom rezervom u moći izračunavanja.

Dokumentacija. Leder dijagrami, kao grafički prikazi, su u toj meri samo-deskriptivni da obično nije neophodna neka dodatna dokumentacija koja bi upotpunjavala opis rada PLC-a i načina na koji su realizovane njegove funkcije. Leder dijagram se uvek može odštampati, a pošto se isti dijagram koristi i kao program, ne postoji opasnost da dokumentacija bude neažurna, što je često slučaj sa dijagramima i šemama relejnih panela (kada inženjer nakon učinjene intervencije ne unese izmenu u relejnu šemu).

Bezbednost. Program PLC-a se ne može promeniti pre nego što je PLC „otključan“ (svaki PLC ima prekidač s ključem - bez ključa nije moguće menjati program).

Mogućnost reprogramiranja. Osobina brzog reprogramiranja PLC-a otvara mogućnost za postizanje neke vrste adaptivnog proizvodnog procesa, gde se program rada menja shodno karakteristikama svakog pojedinačnog proizvoda ili varijacijama u procesu proizvodnje.

4.3.2 Nedostaci

Nova tehnologija. Pokazalo se da je teško promeniti način razmišljanja industrijskih inženjera sa relejne logike na PLC koncept. Međutim, zahvaljujući sve široj primeni, ne samo u domovima i kancelarijama već sve više i u fabričkim halama, računari postaju i sredstvo za postizanje veće produktivnosti proizvodnje.

Aplikacije sa fiksnim programom. Pojedine aplikacije zasnovane su na samo jednoj funkciji koja se veoma retko ili nikada ne menja. U ovakvim slučajevima, zamena postojeće opreme PLC-om ne donosi veliki dobitak, jer se njihova glavna osobenost – mogućnost reprogramiranja – praktično ne koristi. PLC je najbolje rešenje kada su neophodne periodične promene u načinu rada.

Uslovi rada. Ekstremnim uslovi rada, kao što su: visoka temperatura, vlažnost, vibracije, električne smetnje, a koji su karakteristični za pojedine proizvodne procese, mogu uticati na rad PLC-a i ograničiti njegovu primenu i/ili životni vek.

Bezbednost u radu. Kod relejnih sistema uvek postoji tzv. *Stop* prekidač, kojim se u bilo kom momentu može trenutno prekinuti rad sistema (isključenjem napajanja). Pri tome, relejni sistem se automatski ne resetuje kada se napajanje uključi, već zadržava stanje u kome je bio kada je napajanje isključeno. Ovakvo ponašanje se svakako može programski ostvariti i kod PLC-a, tako što će se *Stop* prekidač povezati na jedan od ulaza PLC-a. Međutim, ovakvo rešenje nije bezbedno (ako PLC otkaže, prekidač *Stop* gubi funkciju!) Ovaj nedostatak se može prevazići ugradnjom bezbedonosnih relea na izlazima PLC-a.

4.4 PLC sistem

Na Sl. 4-4 su prikazane četiri osnovne jedinice svakog PLC sistema kao i način na koji su međusobno povezane:

1. **Centralna procesorksa jedinica** (CPU) ili **logička jedinica**. Predstavlja „mozak“ sistema, a sastoji se iz sledeće tri podjedinice:

- *Mikroprocesor*

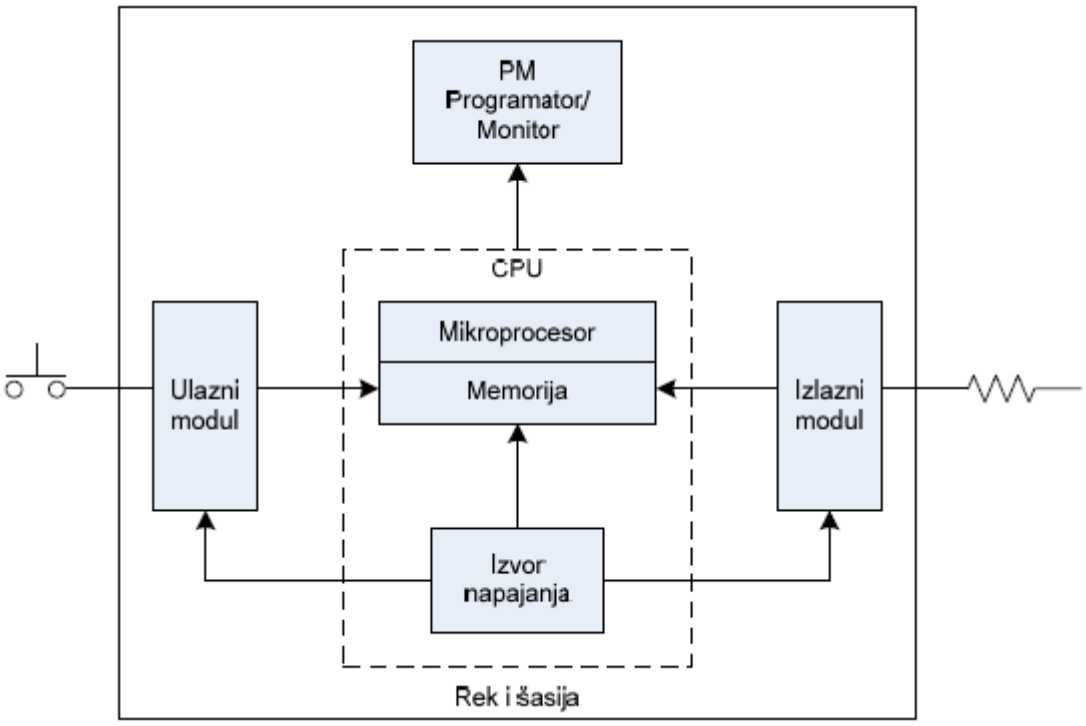
- *Memorija* – za čuvanje sistemskog softvera i korisničkog programa

- *Izvor napajanja* – obezbeđuje napajanje mikroprocesora, memorije, ulaznog i izlaznog modula.

2. **Programator/Monitor** (PM). PM je uređaj koji se koristi za komunikaciju sa PLC-om. Primeri PM-ova su: ručni terminali, industrijski terminali i personalni računari. Sa jedne strane, prema operateru, ovi uređaji poseduju ekran i tastaturu, dok sa druge, prema PLC-u, odgovarajući komunikacioni interfejs za prenos programa, podataka, statusnih informacija ka/iz PLC-a.

3. **U/I moduli.** Ulazni modul poseduje terminale (priključne tačke) na koje se dovode električni signali koje generišu senzori ili prevarači. Izlazni modul poseduje terminale preko koji PLC šalje izlazne signale kojima pobuđuje relee, solenoide, motore, displeje i druge izlazne uređaje ili aktuatore.

4. **Rekovi i šasije.** Delovi PLC sistema CPU, PM i U/I moduli smeštaju se u metalne ormare – tzv. rekove.



Sl. 4-4 PLC sistem.

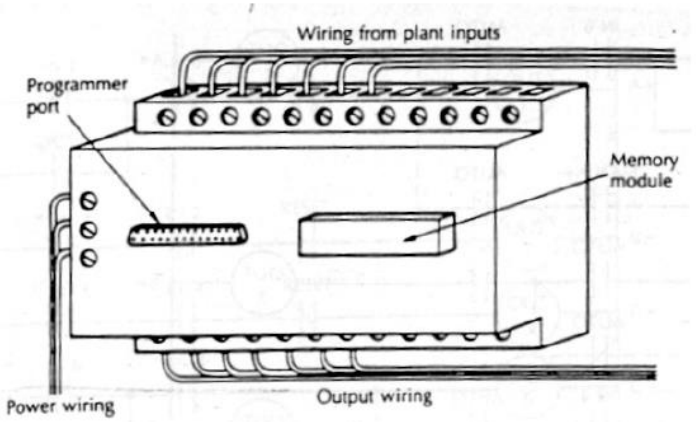
4.4.1 Konstrukcija PLC-a

Razlikuju se dva osnovna načina konstrukcije PLC kontrolera:

(a) kompaktni PLC kontroleri i

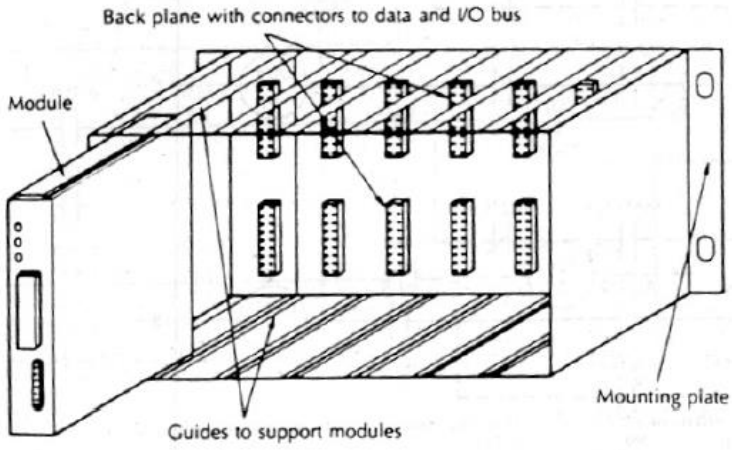
(b) modularni PLC sistemi.

Kompaktni PLC kontroleri su nezavisni, zatvoreni uređaji sa fiksnim brojem ulaza/izlaza, bez mogućnosti proširenja (Sl. 4-5). U jednom kućištu, obično manjih dimenzija, smešetni su: izvor napajanja, procesorska jedinica i ulazni i izlazni modul. Kompaktni PLC kontroleri predstavljaju ekonomično rešenje, predviđeno za upravljanje sistemima i procesima male složenosti. Tipično, poseduju do 16 ulaza i 16 izlaza i memoriju od nekoliko KB.



Sl. 1-5 Kompaktni PLC. (*Uočavaju se priključci za ulaze, izlaze i napajanje, konektor za vezu sa programatorom i slot za memorijski modul*)

Modularni PLC sistemi se sastoje od većeg broja modula koji su smešteni unutar mehaničkog okvira, tj. šasije, koji se zove rek (*rack*) (Sl. 4-6). Rek poseduje veći broj slotova za smeštanje modula. Svaki slot čini par vođica duž gornje i donje stranice reka koje služe za mehaničko učvršćenje modula kao i konektor na zadnjoj ploči reka za priključuje modula na zajedničku magistralu izvedenu na štampanoj ploči zadnje stranice reka. Po pravilu, prvi slot je namenjen modulu izvora napajanja, koji se priključuje na mrežni napon (220Vac) i generiše jednosmerne napone potrebne za rad ostatka sistema. Sledeći, drugi slot se koristi za modul logičke jedinice, tj. procesorski modul koji izvršava korisnički program i upravlja radom ostalih modula. Preostali slotovi se koriste za module specijalne namene, kao što su U/I moduli, memorijski moduli i sl. Ovakav način kontrukcije omogućava lako proširenje sistema. Na primer, ako je potrebno povećati broj ulaza/izlaza dovoljno je ugraditi dodatni U/I modul. Ili, ako zbog povećanih zahteva obrade, postojeći procesorski modul više nije odgovarajući, on se može zameniti novim, moćnijim, a da pri tome ostali moduli ne moraju biti zamenjeni. Broj slotova u jednom PLC reku je, tipično, od 4 do 16. Mogućnost proširenja PLC sistema nije ograničena samo na jedan rek. Uz pomoć posebnih modula za proširenje moguće je povezati dva ili više reka, što omogućuje da se jednim procesorskim modulom upravlja velikim brojem dodatnih modula.



Sl. 4-6 Modularni PLC.

4.4.2 CPU i PM

CPU je „srce“ PLC sistema. Na Sl. 4-7 je prikazana tipična CPU jedinica u spoju sa programatorom/monitorom (PM) oblika ručnog terminala. CPU jedinca može biti veća ili manja od one prikazane na Sl. 4-7, zavisno od veličine i složenosti procesa kojim upravlja.

Jedan od glavnih parametara o kome se mora voditi računa prilikom izbora CPU jedinice jeste veličina interne memorije. Za upravljanje jednostavnim procesom, dovoljan je mali PLC sa ograničenom memorijom; za upravljanje većim sistemom, neophodan je veći PLC, koji podržava naprednije funkcije i poseduje veću količinu memorije. Kod nekih PLC -ova postoji mogućnost da se naknadno ugradi dodatna memorija; kod drugih ta mogućnost ne postoji.

Sl. 4-7 CPU i PM.

Na samoj CPU jedinici obično se mogu naći različiti konektori za kablove koji služe za povezivanje sa drugim PLC-ovima.

Mnoge CPU jedinice poseduju ugrađenu tzv. *backup* bateriju, koja omogućavaju da se u slučaju nestanka eksternog napajanja sačuva učitani korisnički leder program. Za razliku od operativnog sistem PLC-a koji čuva se u permanentnoj memoriji (ROM), korisnički program se čuva u RAM-u, koji, kao što znamo, gubi upisani sadržaj onog momenta kada se napajanje

isključi. Kada PLC detektuje prestanak eksternog napajanja, *backup* baterija se automatski uključuje da bi se obezbedio napon napajanja za RAM i tako sačuvao njegov trenutni sadržaj do dolaska eksternog napajanja.

Na Sl. 4-8(a) je prikazan tipičan programator/monitor (PM) sa CRT ekranom. Tipičan ručni terminal prikazan je na Sl. 4-8(b). PM sa Sl. 4-8(a) može pružiti potpuniji uvid u rad i trenutni status PLC-a, ali zato je ručni terminal prenosiv.

(a) (b)

Sl. 4-8 (a) PM sa CRT ekranom; (b) PM oblika ručnog terminala.

PM se povezuje sa CPU-om pomoću odgovarajućeg kabla. Nakon što je CPU programiran, PM nije više neophodan i veza sa CPU-om se može raskinuti, isti PM se može koristiti za programiranje svih PLC-ova u jednom proizvodnom pogonu.

4.4.3 PLC ulazni i izlazni moduli

PLC pribavlja informacije iz okruženja posredstvom ulaznih, a predaje informacije okruženju putem izlaznih modula. Pristupne tačke (terminali) ulaznog modula primaju signale senzora i pretvarača. Na pristupnim tačkama izlaznog modula generiše se napon za pobudu aktuatora (motori, sklopke, ...) i uređaja za indikaciju (svetiljke, displeji, ...)

UI modul može imati 4, 8, 12 ili 16 terminala. Modul može biti ulazni, izlazni ili kombinovani (U/I) sa podjednakim ili različitim brojem ulaznih i izlaznih terminala (npr. 12 ulaza i 8 izlaza).

Kod manjih sistema, CPU, ulazni i izlazni moduli se smeštaju u isti rek. Kod većih PLC sistema, ulazni i izlazni moduli su smešteni u posebne rekove koji su sa CPU-om povezuju pomoću odgovarajućeg višežičnog kabla.

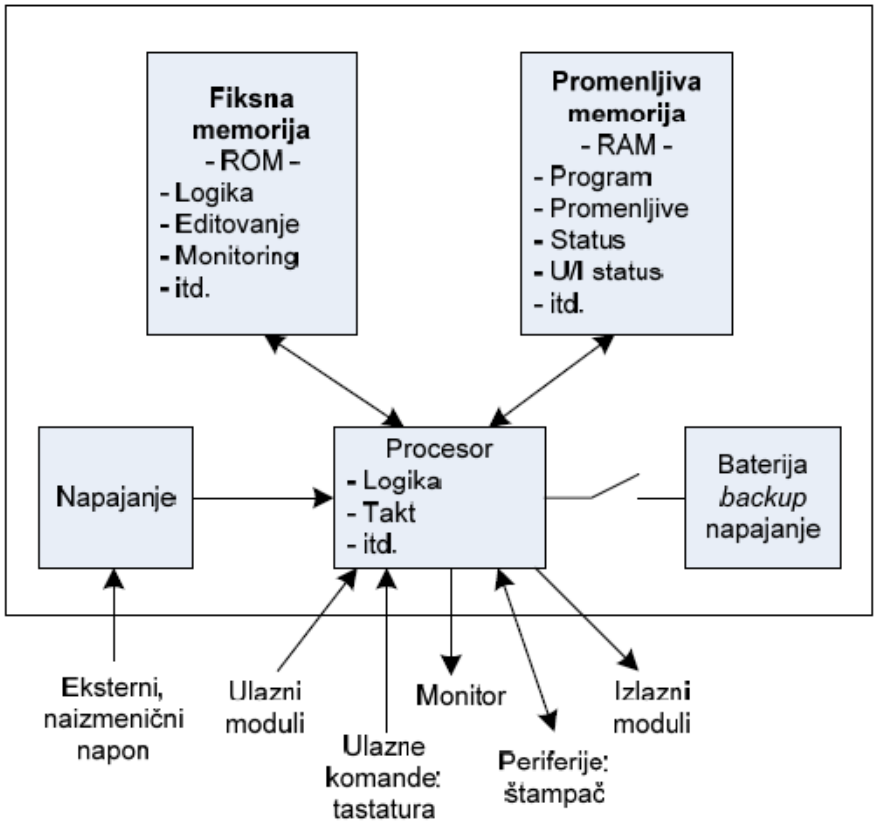
Najvažnija karakteristika U/I modula je opseg napona ili struje koji modul prihvata, odnosno generiše. Naponski i strujni opseg modula mora biti usaglašen sa električnim karakteristikama sistema ili uređaja s kojim se PLC modul povezuje. Da bi se zadovoljili

najrazličitiji zahtevi u pogledu naponskih i strujnih opsega, proizvođači PLC sistema nude palete ulaznih i izlaznih modula deklarisanih za različite naponske/strujne opsege.

Pored diskretnih ulaznih i izlaznih modula (prihvataju i generišu diskretne - digitalne, tj. ON/OFF signala), u upotrebi su i analogni ulazni i izlazni moduli koji prihvataju i generišu analogne signale. Takvi moduli poseduju ugrađene A/D, odnosno D/A konvertore.

4.5 PLC sistem

Bez obzira na veličinu PLC-a, centralna procesorska jedinica, tj. CPU, ili samo *procesorska jedinica*, uvek objedinjuje mikroprocesor i memoriju. Kod većih PLC-ova, CPU sadrži samo mikroprocesor i memoriju, dok kod manjih, dodatno, sadrži U/I interfejs i izvor za napajanje. Takođe, moguće je da CPU sadrži mikroprocesor, memoriju i izvor napajanja, a da je U/I interfejs realizovan eksternim U/I modulima. Jedna takva organizacija, prikazana je blok dijagramom sa Sl. 4-9. Fiksna memorija sadrži program (ili programe) koje je u postavio proizvođač. Ovo su programi *operativnog sistema* PLC-a, koji imaju sličnu namenu, mada sa neuporedivo manjim mogućnostima, kao npr. DOS operativni sistem kod starijih PC računara, Windows kod novijih ili UNIX kod radnih stanica. Programi operativnog sistema su smešteni u posebnom memorijskom integrisanom kolu tipa ROM. Kao što znamo, sadržaj ROM-a se ne može brisati niti menjati, a ostaje nepromenjen i nakon isključenja napajanja. Promenljiva memorija, realizovana pomoću RAM memorijskih čipova, podeljena je na veći broj sekcija koje sadrže korisnički program, programske promenljive, trenutne statusne informacije kontrolera itd.



Sl. 4-9

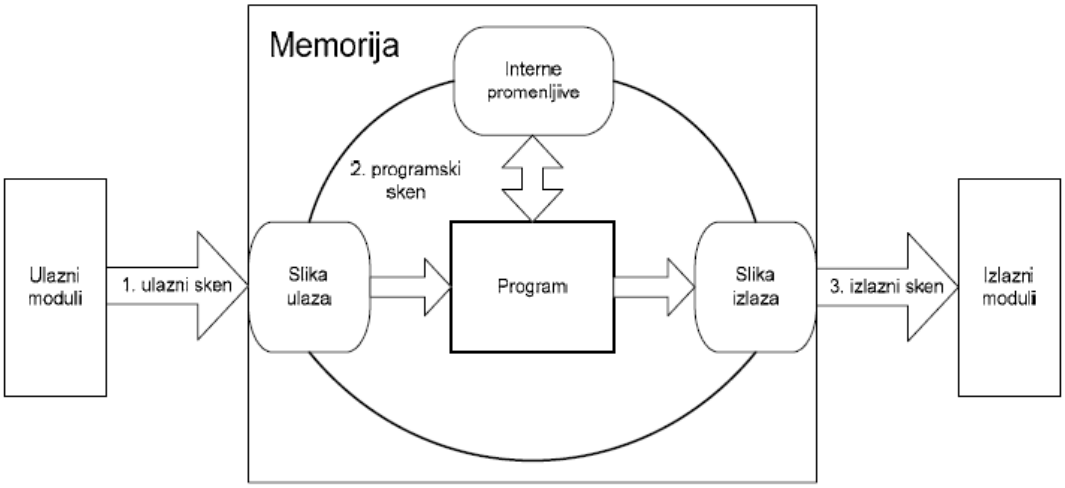
4.5.1 Sken ciklus

Operativni sistem PLC kontrolera je projektovan tačno za određenu vrstu primene. Naime, pretpostavlja se da će u svojoj osnovnoj formi, PLC biti korišćen za realizaciju izvesnih logičkih funkcija koje preslikavaju signale sa senzora u signale koji se prenose na aktuatore. Otuda se od PLC-a očekuje da *periodično očitava* (unosi) signale sa senzora, *izvršava* određen broj aritmetičko-logičkih operacija (u skladu sa zadatom funkcijom) čiji rezultati se *prenose* na izvršne organe ili neke druge indikatorske uređaje. Pored toga, sa istom ili nekom drugom učestanošću, PLC treba da održava *komunikaciju* (razmenjuje podatke) sa nekim drugim računarskim sistemima u mreži. Polazeći od ovog zahteva, operativni sistem PLC kontrolera projektovan je tako da, u toku rada sistema, automatski obezbedi ciklično ponavljanje navedenih aktivnosti (*Sken ciklus*) kao što je to ilustrovano na Sl. 4-10.

Sl. 4-10 Sken ciklus PLC kontrolera.

Sken ciklus započinje sa ***ulaznim skenom*** u okviru koga PLC očitava 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-logičke 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 područja gde se smeštaju interne promenljive. Rezultati obrade se smeštaju u posebno područje memorije – ***slika izlaza***. Važno je istaći da se pri izvršavanju programskih naredbi podaci ne uzimaju direktno sa ulaznih modula, niti se rezultati direktno postavljaju na izlazne module, već program razmenjuje podatke isključivo sa memorijom (Sl. 4-11). Po završetku programskog skena, operativni sistem PLC kontrolera aktivira ***izlazni sken*** u okviru koga se podaci iz *slike izlaza* prenose na izlazne linije (registre izlaznih modula). Na ovaj način stvara se utisak da je PLC sve operacije definisane programom obavio u isto vreme. Četvrti 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 tajmeri i registri, 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 ugrađenog mikroprocesora ulazni i izlazni sken ciklus izvršavaju se u vremenu reda milisekundi (od 0.25ms do 2.56ms). Trajanje programskog skena, svakako zavisi od veličine programa.



Sl. 4-11 Razmena podataka za vreme sken ciklusa.

4.5.2 Osnovne karakteristike CPU jedinice

Kao što je već rečeno, CPU modul sadrži mikroprocesor i memoriju. Mikroprocesor obuhvata aritmetičko-logičku jedinicu (*ALU*), registre i upravljačku jedinicu. U funkcionalnom smislu mikroprocesor PLC-a se bitno ne razlikuje od mikroprocesora bilo kog mikroračunara opšte namene. Međutim, razlika postoji u programskom jeziku; dok se za programiranje mikroračunara koristi asemblerski jezik ili neki viši programski jezik (npr. C), za programiranje PLC-ova se koristi, kao što je ranije rečeno, jezik leder dijagrama. U osnovi, osnovna razlika se ogleda u skupu naredbi koji je odabran tako da se zadovolje osnovni zahtevi u pogledu korišćenja PLC-a.

Osnovne karakteristike procesorskog modula izražavaju se preko sledećih elemenata:

Memorija(RAM) – karakteriše se svojom veličinom, mogućnošću proširenja i konfigurisanja za smeštanja programa ili podataka.

U/I tačke – karakteriše se najvećim brojem lokalnih U/I adresa koje podržava procesor u toku ulaznog i izlaznog skena, kao i mogućnošću 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 (komunikacionog interfejsa) koji podržavaju različite topologije mreža i različite komunikacione protokole.

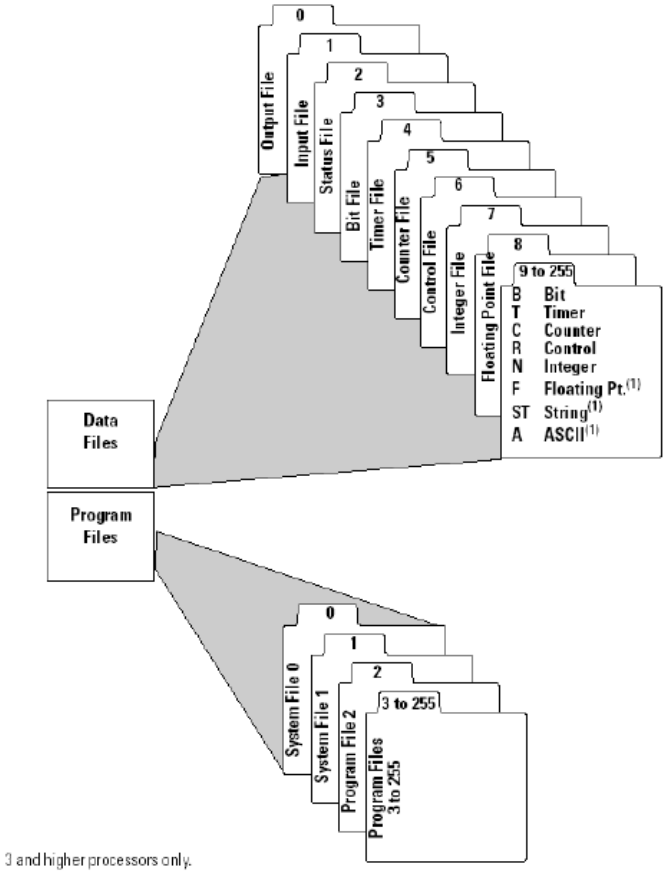
Opcije trajnog pamćenja – odnose se na raspoloživost različitih tipova memorijskih EPROM modula koji obezbeđuju trajno pamćenje podataka.

Performansa – specificira se preko *vremena programskog skeniranja* potrebnog za izvršenje 1Kbajt programa, preko *vremena* potrebnog *za ulazni i izlazni sken*, kao i *vremena izvršavanja jedne bit naredbe*.

Programiranje – specificira se u odnosu na broj podržanih različitih naredbi leder jezika.

4.5.3 Organizacija memorije

Operativni sistem PLC kontrolera, koji realizuje sken cikluse, upravlja i zauzećem RAM memorije, koja je organizovana na poseban način. U principu, RAM memorija se deli na *program files (programske datoteke)* i *data files (datoteke podataka)* Sl. 4-12. 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. To zapravo znači da se jedna aplikacija može razviti na jednom sistemu i zatim u celini preneti i koristiti na drugom sistemu.



Sl. 4-12 Organizacija memorije PLC kontrolera.

4.5.3.1 Programske datoteke

Programske datoteke sadrže informacije o samom kontroleru, glavni korisnički program i potprograme. Svaka aplikacija (procesorska datoteka) mora da ima sledeće tri programske datoteke:

System Program – sistemski program (file 0) - sadrži različite informacije o samom sistemu kao što su tip procesora, 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 PLC 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.

4.5.3.2 Datoteke podataka

Datoteke podataka sadrže podatke koji se obrađuju pomoću naredbi leder programa. Pri tome se pod pojmom *podaci* podrazumevaju konvertovane (numeričke) vrednosti signala koji se preko ulazno/izlaznih modula unose u kontroler, ili se iz kontrolera prenose na izlazne uređaje, 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.

4.5.3.3 Tipovi promenljivih i datoteka

Osnovna karakteristika datoteke podataka je njen *tip*. Kao što je već istaknuto tip datoteke, zapravo ukazuje na vrstu promenljivih koje se u njoj pamte. To nadalje podrazumeva da tip datoteke ujedno određuju i njenu organizaciju, koja zavisi od vrste podatka i usvojenog načina za njegovo prikazivanje u računaru.

Datoteka se označava pomoću *rednog broja*, koji jednoznačno određuje mesto te datoteke u nizu datoteka podataka koje se nalaze u procesorskoj datoteci i *slova* kojim se identifikuje tip datoteke. Prvih devet 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.

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

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

File 5 – *Tip* **C** - *counter* (*brojač*) – 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 formatu pokretnog zareza kao 32-bit brojeve u opsegu (1.1754944e-38 to 3.40282347e+38).

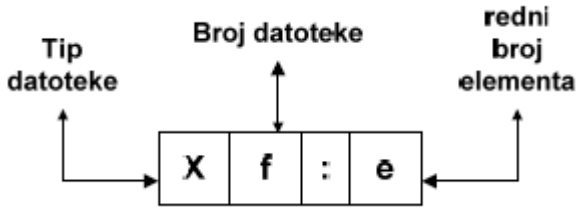
File 9 do file 255 – *Tip* **definiše korisnik** - k*orisničke datoteke* – ove datoteke definiše korisnik kao datoteke tipa **B, T, C, N**.

4.5.3.4 Element datoteke

Osnovna jedinica datoteke je jedan *element*. Svaki elemenat se sastoji iz nekoliko (jedne ili više) *16-bitnih reči*. Broj reči koje čine jedan element zavisi od tipa datoteke, odnosno vrste podataka koji se u nju smeštaju.

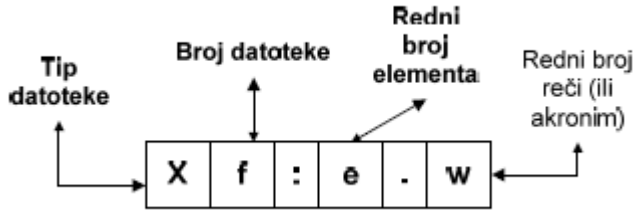
Kao što je već istaknuto, podaci koji su smešteni u datotekama predstavljaju operande (promenljive) koji se koriste u pojedinim programskim naredbama. Ove promenljive se u programu pozivaju preko svojih simboličkih imena koja predstavljaju *logičke adrese*. Pri tome, adrese omogućavaju da se pozove ne samo elemenat u celini, već i njegov deo. To znači da se mogu adresirati pojedine reči u okviru elementa ili pojedini bitovi u okviru reči. Budući da su podaci u izvesnom smislu hijerarhijski organizovani: 1 elemenat sadrži nekoliko reči, a 1 reč 16 bitova, to su i odgovarajuće adrese struktuirane po istom hijerahijskom principu. Pojedine reči i bitovi u nekim datotekama imaju i pridružene akronime (slovne skraćenice), što dodatno olakšava njihovo korišćenje.

Adresa elementa – u principu, svaki element u okviru datoteke se identifikuje pomoću njegovog relativnog položaja u odnosu na početak datoteke (nulti, prvi, drugi, … element). U skladu s tim adresa elementa ima izgled kao na Sl. 4-13.



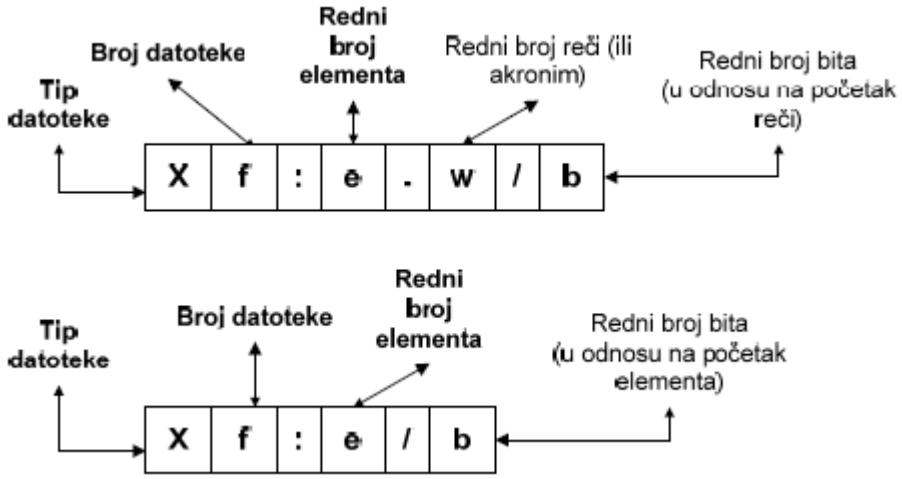
Sl. 4-13 Adresa elementa.

Adresa reči – jedna reč elementa se identifikuje ili pomoću relativnog položaja te reči u okviru elementa, ili pomoću posebnog akronima (ukoliko je isti definisan). Format adrese jedne reči prikazan je na Sl. 4-14.



Sl. 4-14 Adresa reči

Adresa bita – Jedan bit u okviru reči identifikuje se ili preko njegovog relativnog položaja u okviru te reči (*nulti, prvi, drugi, ... bit brojano s desna u levo*) ili preko relativnog položaja u odnosu na početak odgovarajućeg elementa kome pripada reč čiji se bit adresira (Sl. 4-15).



Sl. 4-15 Adresa bita.

4.6 Leder programiranje

Ako se PLC posmatra kao mikroprocesorski sistem, što on i jeste, onda bi se moglo očekivati da se za njegovo programiranje koriste standardni programski jezici. Međutim, ako se pođe od činjenice da je PLC projektovan kao namenski mikroprocesorski sistem za upravljanje i nadzor rada nekog procesa, i da u skladu s 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. Kao što je već ranije istaknuto, PLC je početno razvijen sa idejom da zameni relejne sisteme. To znači da se očekivalo da on realizuje odgovarajuću vremensku sekvencu logičkih operacija. Pored toga, uspešna primena PLC-a u praksi, zahtevala je i da se njegovo programiranje prilagodi tehnici koja je svim korisnicima relejnih sistema dobro poznata. Iz svih ovih razloga, za projektovanje PLC-ova

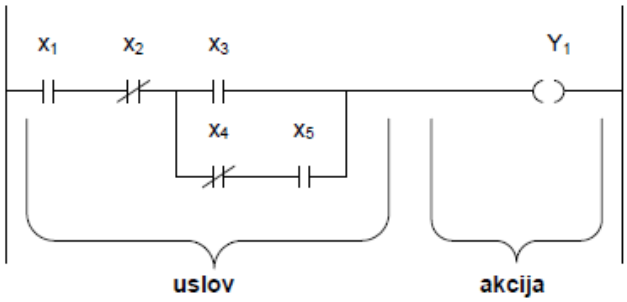
razvijen je programski jezik zasnovan na *leder* (*lestvičastim*) *dijagramima – leder programski jezik.*

4.6.1 Rang

Jedna programska linija leder jezika sastoji se iz niza grafičkih simbola (programskih naredbi) koji predstavljaju različite logičke elemente i druge komponente kao što su tajmeri 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*) (Sl. 4-16).

Uslov – Grafički simboli na levoj strani ranga odnose se ili na stanja signala koji predstavljaju fizičke ulaze PLC-a, i čije su vrednosti tokom ulaznog dela sken ciklusa smeštene u slici ulaza (*input image file*), ili na stanja internih promenljivih, čije su vrednosti smeštene u odgovarajućim datotekama. Svaki simbol predstavlja jednu *unarnu binarnu operaciju* kojoj je pridružena odgovarajuća tablica istinitosti. Uz grafički simbol naznačava se i *adresa* promenljive koja predstavlja operand. Pri ispitivanju istinitosti uslova smatra se da se nad svim simbolima u jednoj liniji (*redna, serijska veza*) obavlja *logička "I" operacija*. To znači da je uslov istinit ukoliko je svaki pojedinačni iskaz istinit. Na levoj strani ranga dozvoljena su i *granjanja* (*paralelene veze*). Pri ispitivanju istinitosti uslova paralelene veze se tretiraju kao *logička "ILI" operacija*. To znači da će iskaz predstavljen nizom paralelnih grana biti istini, ako bar jedna od grana sadrži istinit iskaz. Potrebno je da se istakne da leva strana ranga može biti formirana i tako da na njoj nema ni jednog simbola. U tom slučaju smatra se da je uslov koji se na taj način definiše uvek istinit.



Sl. 4-16 **Leder rang**

Akcija – Grafički simboli na desnoj strani ranga odnose se ili na fizički izlaz (promenljive smeštene u slici izlaza (*output image file*), koje će biti prenete na izlaze kontrolera u toku izlaznog dela sken ciklusa) ili na interne promenljive, čije su vrednosti smeštene u odgovarajućim datotekama. Svaki simbol predstavlja jednu *naredbu* koja se izvršava ako je uslov na levoj strani istinit. Uz simbol se naznačava i adresa promenljive čija se vrednost menja prilikom izvršavanja naredbe, ili koja na bilo koji drugi način učestvuje u realizaciji naredbe (npr. otpočinjanje ili zaustavljanje neke aktivnosti, skok na neki drugi rang, poziv

potprograma itd.). *Serijska veza* na desnoj strani ranga *nije* dozvoljena, dok *paralelna veza* označava da se više različitih naredbi izvršavaju kao rezultat ispitivnja istinitosti jednog istog uslova.

U literaturi je uobičajeno da se i simboli koji označavaju *uslov* i simboli koji označavaju *akciju* označavaju kao ***naredbe***. Otuda je neophodno da se istakne suštinska razlika između *naredbi uslova* i *naredbi akcije*. Naime, izvršavanje *naredbi uslova* obavlja se tako što se u zavisnosti od vrednosti operanda, prema pridruženoj tablici istinitosti, naredbi ***dodeljuje vrednost*** **(0** ili **1)**. Dakle, *naredbe uslova* se izvršavaju u svakom sken ciklusa i rezultat njihovog izvođenja je *vrednost naredbe*. Za razliku od toga *naredbama akcije* se ili ***dodeljuje vrednost nekoj promenljivoj*** ili ***izvršava neka druga aktivnost***. Ove naredbe se izvršavaju samo ako je *uslov* koji im prethodi *istinit* (dodeljena mu je vrednost 1). Pri tome se samim *naredbama akcije* ***ne dodeljuje nikakva vrednost***.

Leder program se izvršava u toku programskog dela sken ciklusa i to tako što se obrađuje rang po rang po redosledu kako su oni poređani u leder dijagramu. U svakom rangu ispituje se istinitost uslova i ukoliko je uslov istinit izvršavaju se odgovarajuće naredbe u desnom delu ranga. To znači da promenljive u desnom delu ranga mogu menjati svoju vrednost samo jedanput u toku sken ciklusa, i to upravo onda kada se odgovarajući rang ispituje. Potrebno je zapaziti da ukoliko se promenljiva na desnoj strani ranga odnosi na fizički izlaz, vrednost izlaza neće biti promenjena u istom trenutku vremena. Naime, za vreme programskog skena menjaju se samo vrednosti promenljivih smeštenih u *slici izlaza*. Tek kasnije, za vreme izlaznog dela sken ciklusa, sve promenljive iz slike izlaza biće prenete na odgovarajuće izlazne linije. Ista stvar važi i za ulazne promenljive. Drugim rečima, za vreme programskog skena ispitivanje istinitosti uslova odnosi se na vrednosti promenljivih u *slici ulaza*, koje su tu upisane za vreme ulaznog dela sken ciklusa koji je prethodio programskom skenu, a ne na trenutne vrednosti promenljivih na ulaznim linijama. Naravno, svi uslovi i naredbe koji su vezani za interne promenljive izvršavaju se u trenutku skaniranja pojedinog ranga.

4.6.2 *Bit naredbe*

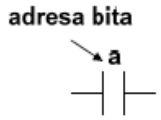
Bit naredbe su, kao što samo ime kaže naredbe čiji su operandi *bitovi*. S gledišta lokacije operanada, to znači da se oni najčešće nalaze u datoteci 3 (bit file - B), digitalnim ulaznim ili izlaznim datotekama (input image file 1 ili output image file 0) ili u korisničkim datotekama bit tipa. Pored toga, adresirani operand može da se nalazi i u bilo kojoj drugoj datoteci u okviru koje je moguće adresirati pojedini bit. 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*).

4.6.2.1 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)

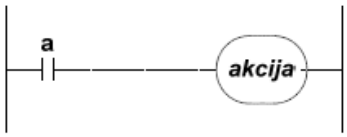
Grafički simbol



Tablica istinitosti

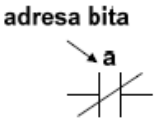
Stanje bita "a"	Vrednost XIC naredbe
0	False
1	True

Položaj u rangu



XIO - Examine if open (ispitivanje da li je kontakt otvoren)

Grafički simbol



Tablica istinitosti

Stanje bita "a"	Vrednost XIC naredbe
0	True
1	False

Položaj u rangu

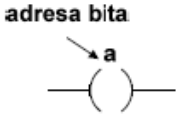
Nazivi ove dve naredbe potiču od ispitivanja binarnih signala koji dolaze sa prekidačkih kola. U tom smislu, *XIC* naredba se odnosi na *normalno otvoren prekidač*, dok se *XIO* naredba odnosi na *normalno zatvoren prekidač*.

4.6.2.2 Bit naredbe za postavljanje vrednosti izlaza

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

OTE - Output energize (pobuđivanje izlaza)

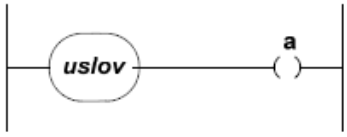
Grafički simbol



Akcija

Ako je vrednost uslova	Bit na adresi "a" dobija vrednost
True	1 (set)
False	0 (reset)

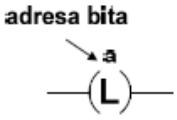
Položaj u rangu



Potrebno je da se zapazi da se ovom naredbom 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)

Grafički simbol



Akcija

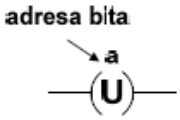
Ako je vrednost uslova	Bit na adresi "a" dobija vrednost
True	1 (set)
False	nepromenjena

Položaj u rangu

OTL naredbom se adresirani bit može isključivo postaviti na 1. Naime, za razliku od *OTE* naredbe kojom se vrednost bita može postavljati na 0 ili 1 svaki put kad se rang skenira, 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)

Grafički simbol



Akcija

Ako je vrednost uslova	Bit na adresi "a" dobija vrednost
True	0 (reset)
False	nepromenjena

Položaj u rangu

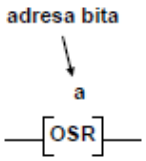
OTU naredbom se adresirani bit može isklju čivo postaviti na 0. Pri tome, vrednost bita se postavlja (lečuje) na 0 u prvom skenu u kome je *uslov* ispunjen. Nakon toga ova naredba postaje neosetljiva na vrednost *uslova*.

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

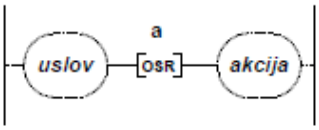
4.6.2.3 Bit triger naredba

OSR - One-shot rising (uzlazna ivica)

Tablica istinitosti



Ako je uslov	Vrednost naredbe je	Bit na adresi "a" dobija vrednost
True	True, u sken ciklusu u kome uslov postaje istinit	set (1)
False	False, u ostalim sken ciklusima	reset (0)



O*SR* naredba omogućava da se obezbedi izvodjenje neke akcije *samo jedanput*. Ova naredba je specifična po tome što istovremeno pripada i kategoriji *uslova* i kategoriji *akcije*. Naime ova naredba se postavlja u rangu *između* dela koji predstavlja *uslov* i dela koji predstavlja *akciju*. 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 čija je adresa pridružena toj naredbi dodeljuje vrednost 1 (po čemu se ova naredba svrstava i u kategoriju akcija). Obe ove vrednosti ostaju nepromenjene do sledećeg sken ciklusa, kada naredba dobija vrednost *neistinit*, dok se adresirani bit postavlja na vrednost 0 ili 1 u zavisnosti od vrednosti uslova. U narednim sken ciklusima vrednost naredbe ostaje nepromenjena sve dok se u *uslovu* (koji predstavlja ulaz u *OSR*) ponovo ne detektuje prelaz "neistinit/istinit".

Potrebno je istaći da bit čija je adresa pridružena ovoj naredbi ne predstavlja vrednost naredbe. Naime, ovaj bit se koristi kao interna promenljiva i služi za pamćenje *vrednosti uslova* koji prethodi *OSR* naredbi. Vrednost ovog bita je 1 ako je *uslov istinit*, odnosno 0 ako je *uslov neistinit*. U tom smislu, sa aspekta dodeljivanja vrednosti bitu čija se adresa navodi u *OSR* naredbi, ova naredba je identična sa *OTE* naredbom. Navedeni bit se može nalaziti u bilo kojoj bit-adresibilnoj datoteci ***izuzev*** datoteke ulaza i izlaza.

Vrednost koju dobija *OSR* naredba koristi se kao *uslov* za izvođenje naredbe *akcije* koja se nalazi na desnoj strani ranga (neposredno iza *OSR* naredbe). Shodno tome, naredba *akcije* biće izvršavana ***po jedanput*** pri svakom prelazu *uslova* "*neistinit/istinit*".

Iza *OSR* naredbe se može nalaziti samo jedna naredba akcije.

4.7 Metod konačnih automata

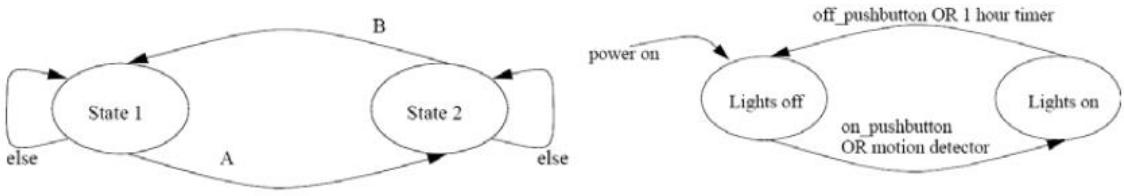
Većina sistema u oblasti automatskog upravljanja su sekvencijalni po svojoj prirodi. Tokom noramalnog rada, sekvencijalni sistem prolazi kroz više koraka, tj. faza ili stanja. U svakom stanju, sistem se ponaša na drugačiji način. Na primer, zamislimo semafor za regulisanje saobraćaja. Jedno stanje semafora može biti ono kada je dozvoljen saobraćaj u jednom, a drugo kada je dozvoljen saobraćaj u drugom smeru. Svakom stanju semafora odgovara jedna specifična kombinacija upaljenih crvenih, žutih i zelenih svetala, a stanja se izmenjuju po unapred utvrđenom redosledu, tj. sekvenci. Ova sekvenca može biti fiksna (nepromenljiva), ali se može i menjati pod dejstvom nekih specifičnih ulaznih uslova. Na primer, ako na semaforu postoji ″pešačko dugme″, tada će redosled u kome se izmenjuju semaforska svetla svakako zavisiti i od toga da li je dugme pritisnuto ili ne. Dakle, pod stanjem se može smatrati vremenski interval u kojem su izlazi sistema stabilni (ne menjaju se) i u kome sistem reaguje na ulaznu pobudu na način specifičan za to stanje.

Sekvencijalni sistemi su po pravilu složeni, a njihovo projektovanje je otežano obiljem detalja o kojima treba voditi računa. Zbog toga, direktno projektovanje, gde se na osnovu polazne specifikacije, a bez neke veće razrade, odmah pristupa implementaciji (pisanju programa, ili crtanju leder dijagrama) ima ograničen domet, a dobijena rešenja često nisu ispravna ili zahtevaju naknadno temeljno testiranje i ispravljanje učinjenih grašaka. Iz tog razloga, razvijeno je više, u osnovi srodnih, metoda projektovanja sekvencijalnih sistema kod kojih je naglasak upravo na razradi problema sa ciljem da se, pre same implementacije, detaljnom analizom identifikuju karakteristike sistema, kao što su stanja, prelazi i događaji, na osnovu kojih se kreira apstraktni model ponašanja sistema koji se potom koristi za neposrednu implementaciju. Jedan od takvih metoda je upravo metod konačnih automata. Ovaj metod se primenjuje ne samo za opis ponašanja sistema automatskog upravljanja, već i

u mnogim drugim oblastima, kao što je projektovanje upravljačkih jedinica digitalnih sistema, ili projektovanje konkurentnog softvera.

Model konačnog automata zasnovan je na konceptu *stanja* i *prelaza* između stanja. U svakom trenutku, automat je u jednom od konačnog broja svojih stanja. Pod dejstvom događaja ili uslova, a u zavisnosti od tekućeg stanja i trenutne vrednosti ulaza, automat prelazi u novo stanje i generiše odgovaraju će izlaze. Konačni automat se može predstaviti dijagramom (grafom) stanja. U ovom dijagramu, krugovima su predstavljena stanja, a strelicama (granama) prelazi između stanja.

Dijagram stanja prikazan na Sl. 4-17(a) ima dva stanja State 1 i State 2. Ako je sistem u stanju State 1 i desi se događaj A, sistem prelazi u stanje State 2, inače ostaje u stanju State 1. Slično, ako je sistem u stanju State 2 i desi se događaj B, sistem se vraća u stanje State 1, inače ostaje u stanju State 2. Grane na Sl. 4-17(a) označene sa *else* važe u slučajevima kada ni jedan uslov naveden na izlaznim granama iz dataog stanja nije ispunjen. Uobičajeno je da se ove grane ne crtaju, već se podrazumevaju, tj. ako ni jedan uslov koji vodi sistem u neko drugo stanje nije ispunjen, sistem ostaje u zatečenom stanju.



(a) (b)

Sl. 4-17. (a) Dijagram stanja sa dva stanja; (b) dijagram stanja kontrola osvetljenja.

Dijagram stanja sa Sl. 4-17(a) može se iskoristiti za modelovanje ponašanja kontrolera osvetljenja, kao što je prikazano na Sl. 4-17(b). Stanjima su sada data imena koja ukazuju na režim rada sistema. U stanju *Light off* svetlo je ugašeno, a u stanju *Ligth on* svetlo je upaljeno. Događaj koji prevodi sistem iz stanja *Light off* u stanje *Light on* je ″prekidač uključen ili detektovan pokret″. Događaj koji gasi svetlo, tj. prevodi sistem iz stanja *Ligth on* u stanje *Light off* je ″isteklo vreme od 1h ili prekidač isključen″. Strelica usmerena ka stanju *Light off* i označena sa *reset* ukazuje na početno stanje sistema, tj. na stanje u koje će sistem biti postavljen kada počne sa radom.

4.8 Naredbe za merenje vremena i prebrojavanje događaja – tajmeri i brojači

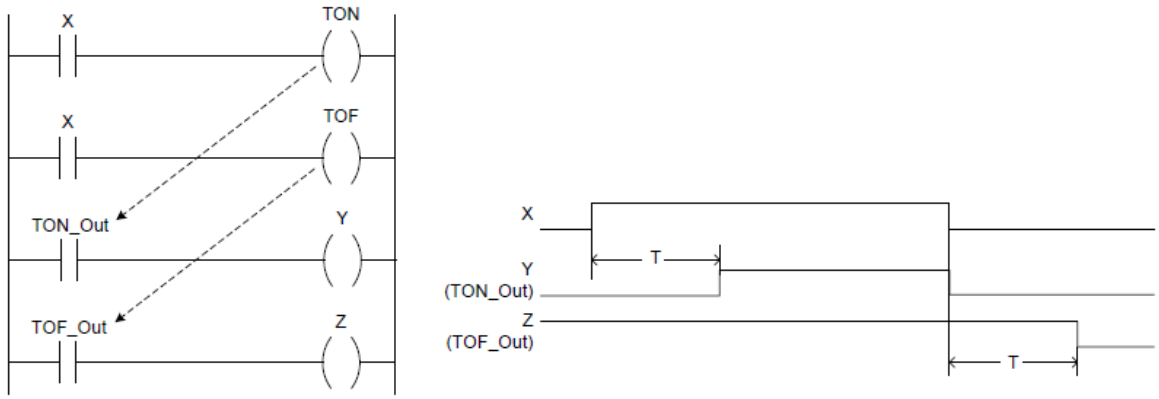
Prilikom upravljanja ili nadzora procesa često je potrebno da se neka aktivnost otpočne ili zaustavi posle određenog vremenskog perioda, ili da se ponovi određeni broj puta. U tom smislu neophodno je da PLC kontroler koji će se koristiti za upravljanje procesom pruži mogućnost za merenje vremena i prebrojavanje događaja. Prebrojavanje događaja obavlja ***brojač*** (***counter***), koji nakon registrovanja unapred zadanog broja događaja generiše odgovarajući signal. Merenje vremena ostvaruje se pomoću ***časovnika, tj. tajmera*** (***timer***). U suštini tajmer izražava vreme kao umnožak određenog osnovnog intervala (*vremenska baza*) .

To zapravo znači da tajmer radi kao brojač protoka osnovnih intervala i da nakon isteka određenog, unapred zadanog intervala vremena, generiše odgovarajući signal.

4.8.1 Naredbe za merenje vremena

PLC naredbe za merenje vremena su, posle bit naredbi, najčešće korišćene naredbe u leder programiranju. Podrška za programsko merenje vremena, u vidu specijalizovanih, tzv. tajmerskih naredbi, postoji kod svih danas raspoloživih familija PLC kontrolera. Iako način rada i mogućnosti ovih naredbi zavise od tipa PLC kontrolera, uopšteno govoreći, u leder programiranju se koriste tri vrste tajmera: *delay_on* tajmer (ili TON), *delay_off* tajmer (ili TOF) i RTO tajmer.

Tajmeri su naredbe akcija. Uslov koji prethodi tajmeru upravlja tajmerom tako što mu omogućava ili brani rad. Tajmer, za vreme dok je omogućen, odbrojava osnovne vremenske intervale i, nakon dostizanja zadate vrednosti, postavlja određeni bit (ili bitove) iz interne memorije PLC kontrolera, koji se mogu koristiti u drugim rangovima za formiranje uslova. Sl. 4-18 ilustruje razliku između *delay-on* i *delay-off* tajmera. TON je omogućen dok je uslov tačan, a TOF dok je uslov netačan. TON se isključuje trenutno, a uključuje po isteku zadatog vremena, T. Nasuprot tome, TOF se uključuje trenutno, a isključuje po isteku zadatog vremena, T. (Ovde se pod uključivanjem/isključivanjem smatra setovanje/resetovanje bita koji tajmer kontroliše – TON_ Out, odnosno TOF_Out). Drugim rečima, TON kasni uključivanje, a TOF isključivanje. Na primer, TON se može iskoristiti da odloži po četak nekog proizvodnog procesa za vreme potrebno da se komora zagreje; TOF se može iskoristiti da ventilator, koji hladi komoru, ostane uključen neko zadato vreme nakon što je sistem isključen.



(a) (b)

Sl. 4-18 Delay-on i delay-off tajmer: (a) leder dijagram; (b) vremenski dijagram.

TON i TOF tajmeri uvek počinju svoj rad ″od nule″, tj. promena uslova na vrednost koja brani rad tajmera ujedno i resetuje tajmer tako da kada sledeći put tajmer bude pušten u rad, njegova početna vrednost biće 0. Za razliku od toga, RTO tajmer (ili, retencioni tajmer) sumira (akumulira) vreme za koje je omogućen. Kada je ponovo omogućen, RTO nastavlja

odbrojavanje osnovnih vremenskih intervala počev od vrednosti koju je imao u trenutku kada je zaustavljen. Za resetovanje RTO tajmera koristi se posebna naredba.

4.8.1.1 Parametri tajmera

Pri korišćenju tajmera neophodno je da se definišu sledeći parametri.

Vremenska baza (*time base*) određuje dužinu *osnovnog intervala vremena*.

Zadata vrednost (*preset value* - PRE) je vrednost kojom se definiše željeni broj osnovnog intervala vremena (čime se određuje ukupno vreme koje časovnik treba da izmeri) koje tajmer treba da registruje pre nego što se generiše signal koji označava da je tajmer završio.

Zadata vrednost za časovnik može da se kreće u intervalu od 0 do +32767, što odgovara vremenu od maksimalno 327.67 s (za vremensku bazu 0.01 s), odnosno maksimalno 32767 s, ili približno 9h i 6min (za vremensku baz 1 s).

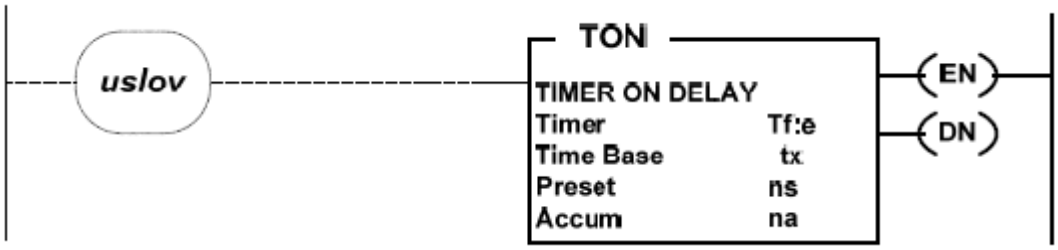
Akumulirana vrednost (***accumulated value* - ACC**) predstavlja broj osnovnih vremenskih intervala koje je časovnik izbrojao u nekom trenutku. Kada akumulirana vrednost postane veća ili jednaka od zadate vrednosti časovnik, odnosno brojač, završavaju svoj rad.

4.8.1.2 Naredbe časovnika

Kao što je već rečeno naredbe tajmera su naredbe *akcije*, što znači da se nalaze na desnoj strani ranga u leder programu. Postoje tri tipa naredbi kojima se realizuju tri vrste tajmera, i jedna naredba kojom se stanje tajmera resetuje.

Potrebno je istaći da se sam tajmer i način njegovog rada definiše preko naredbe koja se uvrštava u leder program. Drugim rečima, kad se u program stavi jedna od moguće tri naredbe i u njoj naznači adresa tajmera u odgovarajućem formatu, onda operativni sistem PLC kontrolera sam zauzme naznačeni element (tri reči) u datoteci koja je navedena u adresi i popuni ga odgovarajućim sadržajem.

- ***Timer on-delay (TON)***



Kao što je već rečeno, stavljanjem ove naredbe u leder program automatski se definiše prva vrsta tajmera i zauzimaju tri reči koje čine elemenat ***e*** u datoteci časovnika ***f***. Prilikom formiranja naredbe specificaraju se i vremenska baza (tx) i zadata vrednost (ns).

TON naredba započinje rad tajmera (prebrojavanje osnovnih vremenskih intervala) za vreme onog programskog sken ciklusa u kome *uslov* u rangu u kome se naredba nalazi prvi put

postaje *istinit* (prelaz *neistini/istinit* – uzlazna ivica). 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 zadatu vrednost, tajmer prekida svoj rad i postavlja DN bit na 1. Pri tom, ako u nekom sken ciklusu *uslov* postane *neistinit*, tajmer prekida svoj rad i akumulirana vrednost se postavlja na 0, bez obzira da li je tajmer pre toga izmerio zahtevano vreme ili ne.

Bitovi stanja časovnika menjaju se u toku programskog sken ciklusa na sledeći način:

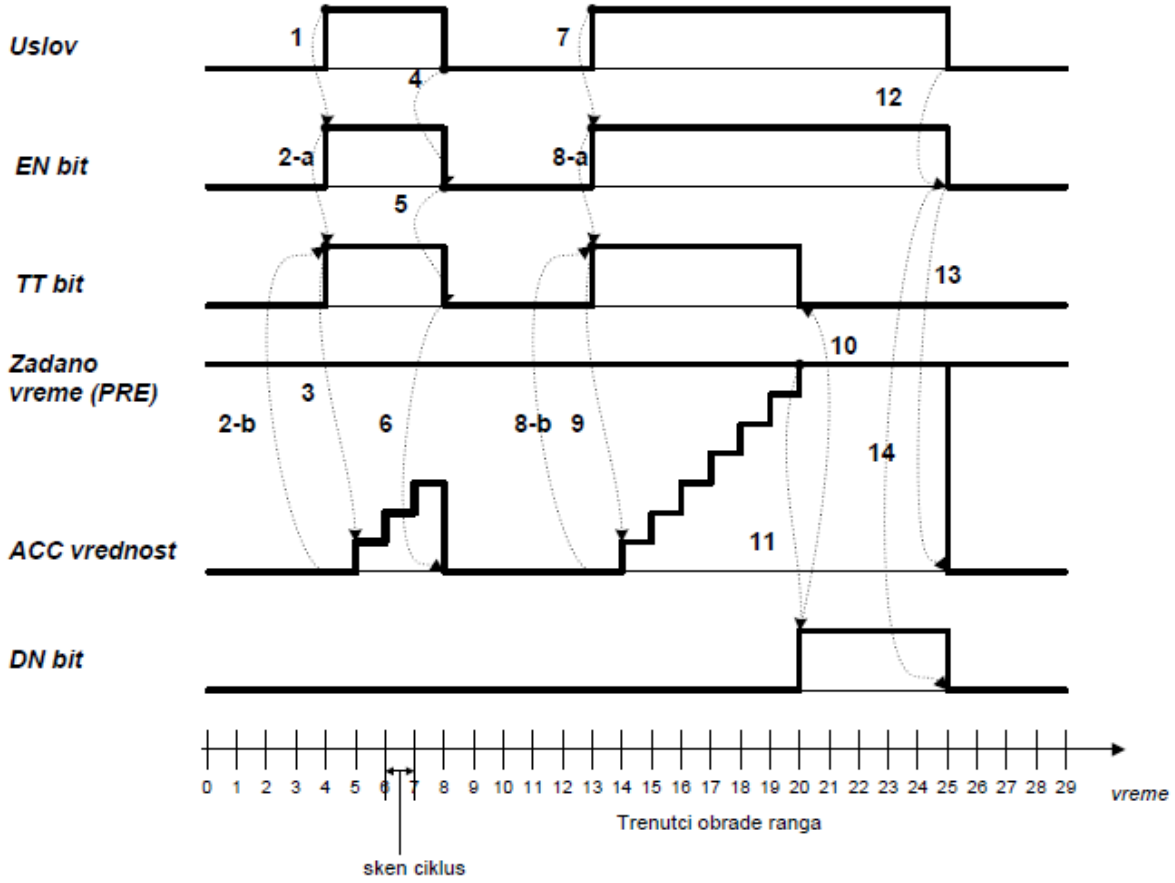
DN - Timer done bit se postavlja na 1 kada je ACC≥PRE. On se resetuje na 0 kad *uslov* u rangu postane *neistinit*.

EN - Timer enable bit se postavlja na 1 kada je *uslov* u rangu *istinit* i resetuje na 0 kada *uslov* postane *neistinit*.

TT - Timer timing bit se postavlja na 1 kada je *uslov istinit* i ako je ACC≤PRE. On se resetuje na 0 kada *uslov* postane *neistinit* ili kada se DN bit postavi na 1, odnosno kada se završi merenje vremena.

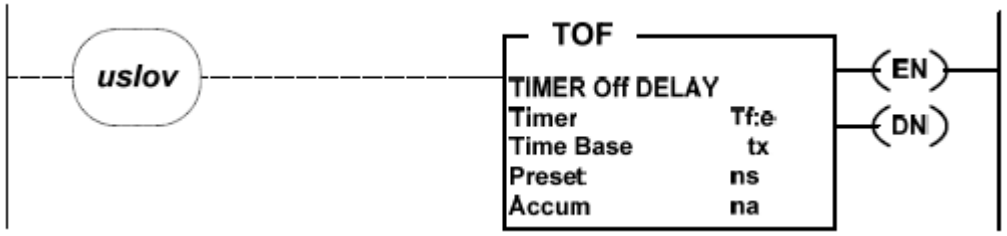
U vezi sa radom tajmera potrebno zapaziti nekoliko činjenica. Pre svega **tajmer radi samo dok je** *uslov istinit* (signal na ulazu u časovnik je u stanju "*on*"). Istinitosnu vrednost uslova pokazuje bit EN. Drugim rečima, ovaj bit ima vrednost 1 onda kada je *uslov istinit* i to označava da je rad tajmera *omogućen* (*enable*). Kad je *uslov neistinit*, EN bit ima vrednost 0, što znači da je rad tajmera *onemogućen*. Međutim, činjenica da EN bit ima vrednost 1 ne mora da znači da tajmer zaista i radi, jer je on mogao i da završi rad zbog isteka zadatog vremena, a da pri tome *uslov* i nadalje ostane *istinit*. **Rad tajmera** inicira TT bit. Naime, taj bit je postavljen na 1 za sve vreme za koje *tajmer aktivno meri vreme* (*timer timing*), i postavlja se na 0 kada tajmer ne radi. Konačno, kada je vrednost DN bita 1, onda to znači da je tajmer *završio* (*done*) svoj posao, tj. izmerio zadato vreme. Pri tom, DN bit ne govori o tome **kada** je tajmer završio sa poslom, jer će on ostati na vrednosti 1 sve dok *uslov* ne postane *neistinit.* Vremenski dijagram rada tajmera ilustrovan je na Sl. 4-19.

Stanje tajmera se može resetovati posebnom *RES naredbom*, o čemu će kasnije biti više reči.



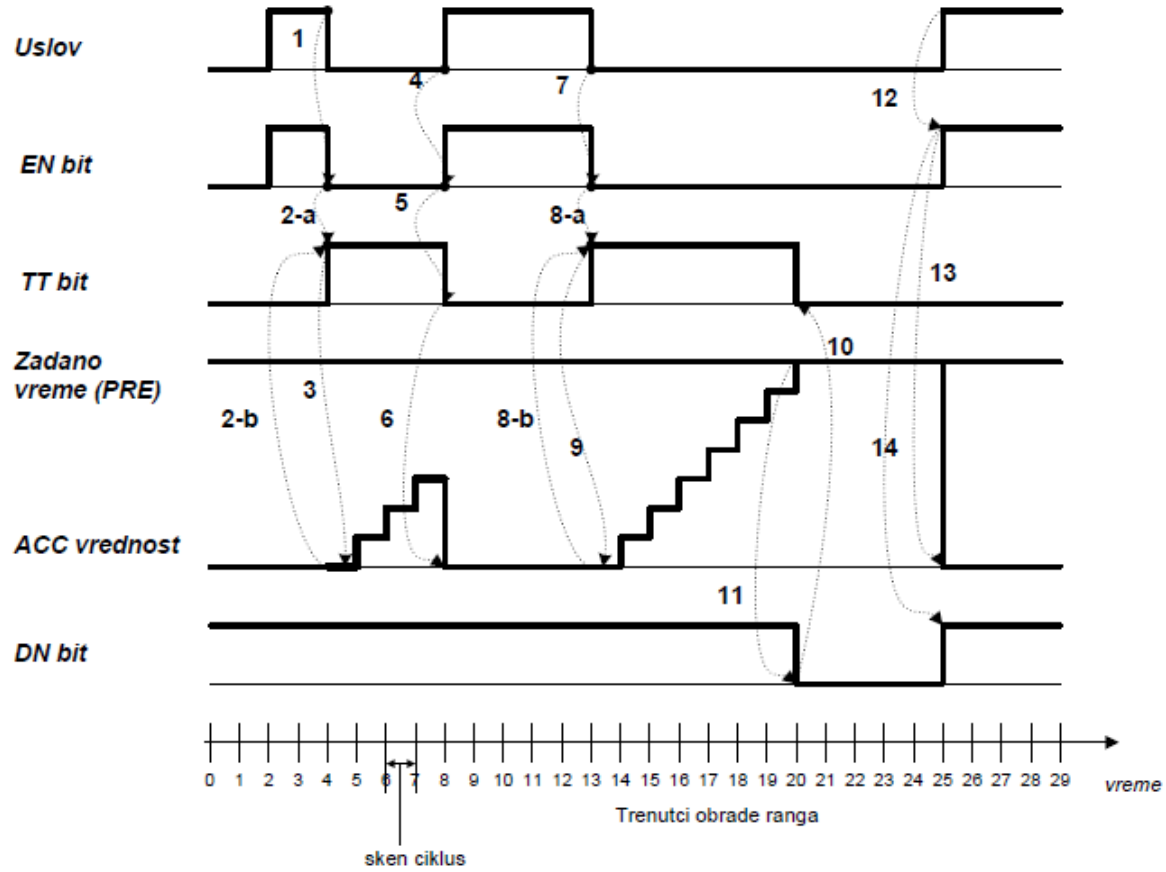
Sl. 4-19 Vremenski dijagram izvršavanja TON naredbe.

- ***Timer off-delay (TOF)***



Ovom naredbom se definiše druga vrsta tajmera i zauzimaju tri reči koje čine elemenat ***e*** u datoteci tajmera ***f***. Prilikom formiranja naredbe specificaraju se i vremenska baza (***tx***) i zadata vrednost (***ns***). Akumulirana vrednost se automatski postavlja na 0.

TOF naredba započinje rad tajmera za vreme onog programskog sken ciklusa u kome uslov u rangu u kome se naredba nalazi prvi put postaje **neistinit** (prelaz istini/neistinit – silazna ivica). U svakom sledećem sken ciklusu, sve dok je uslov *neistinit* časovnik vrši ažuriranje akumulirane vrednosti (ACC) u skladu sa proteklim vremenom između dva ciklusa. Kada akumulirana vrednost dostigne zadatu vrednost, tajmer prekida svoj rad. Pri tom, ako u nekom sken ciklusu uslov postane istinit, tajmer prekida svoj rad i akumulirana vrednost se postavlja na 0, bez obzira da li je tajmer pre toga izmerio zahtevano vreme ili ne.



Sl. 4-19 Vremenski dijagram izvršavanja TOF naredbe

Bitovi stanja TOF tajmera menjaju se u toku programskog sken ciklusa na sledeći način:

DN - Timer done bit se postavlja na 1 kada je uslov istinit. On se resetuje na 0 kada je uslov neistinit i pri tome je ACC≥PRE.

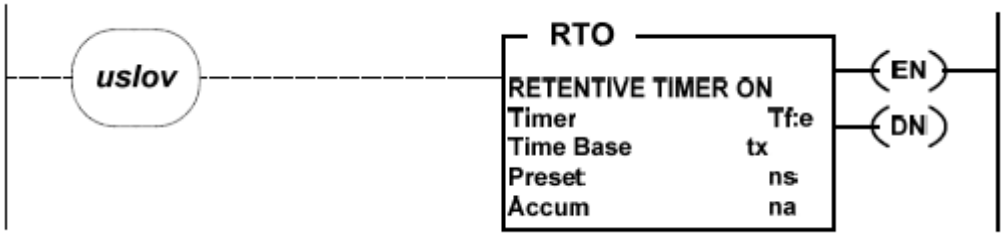
EN - Timer enable bit se postavlja na 1 kada je uslov istinit, i resetuje na 0 kada je uslov neistinit.

TT - Timer timing bit se postavlja na 1 kada je uslov neistinit i pri tome je ACC≤PRE. One se resetuje na nulu kada uslov postane istinit ili kada se DN bit resetuje.

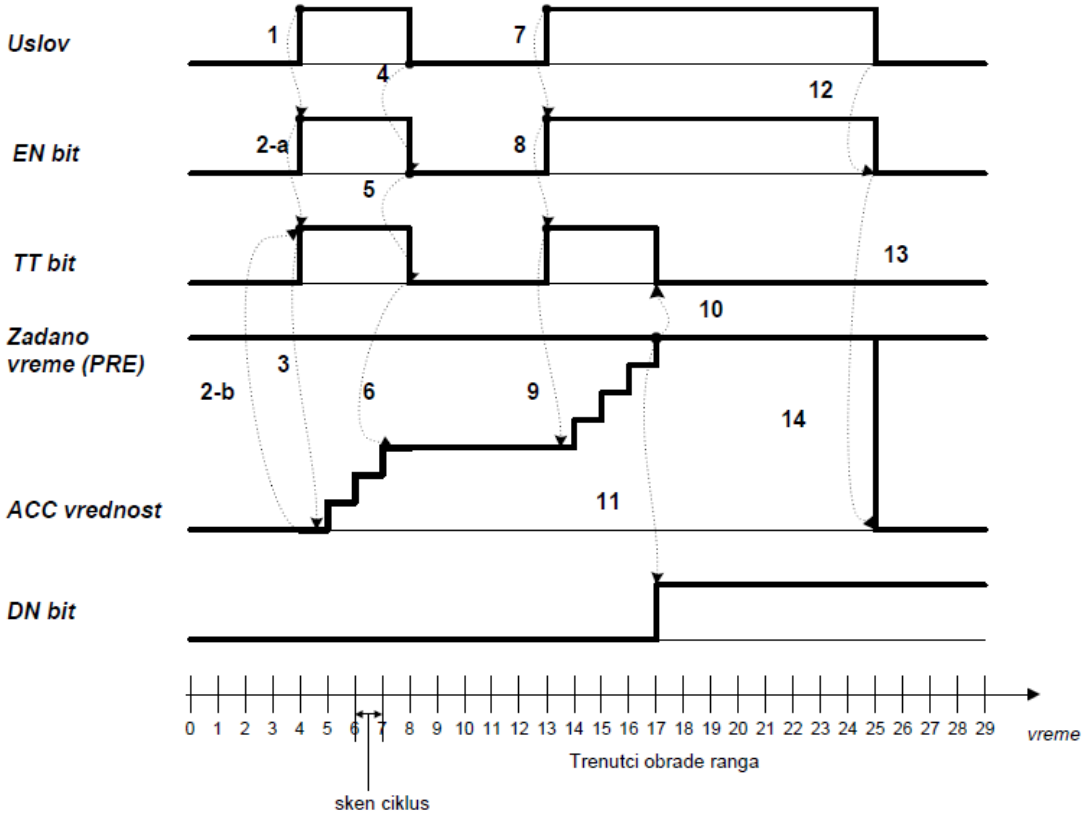
U vezi sa radom TOF tajmera potrebno je da se zapazi nekoliko činjenica. Pre svega tajmer radi samo dok je uslov neistinit (signal na ulazu u tajmer je u stanju "off") . Istinitosnu vrednost uslova pokazuje EN, ali za razliku od TON naredbe, ovde on onemogućava rad tajmera. Drugim rečima, ovaj bit ima vrednost 1 onda kada je uslov istinit i to označava da je rad tajmera onemogućen. Kada je uslov neistinit, EN bit ima vrednost 0, što znači da je rad tajmera omogućen. Međutim, činjenica da EN bit ima vrednost 0 ne mora da znači da tajmer zaista i radi, jer je on mogao i da završi rad zbog isteka zadatog vremena, a da pri tom uslov i nadalje ostane neistinit. Rad tajmera inicira TT bit. Naime, taj bit je postavljen na 1 za sve vreme za koje tajmer aktivno meri vreme (timer timing), i postavlja se na 0 kada tajmer ne radi. Konačno, kada je vrednost DN bita 0, onda to znači da je tajmer završio (*done*) svoj posao, tj. izmerio zadato vreme. Pri tom, DN bit ne govori o tome kada je časovnik završio sa

poslom, jer će on ostati na vrednosti 0 sve dok uslov ne postane istinit. Vremenski dijagram rada časovnika ilustrovan je na Sl. 4-19.

- ***Retentive Timer (RTO)***



Ovom naredbom se definiše treća vrsta tajmera i zauzimaju tri reči koje čine elemenat ***e*** u datoteci časovnika ***f***. Prilikom formiranja naredbe specificaraju se i vremenska baza (***tx***) i zadata vrednost (***ns***). Akumulirana vrednost se automatski postavlja na 0.



Sl. 4-20 Vremenski dijagram izvršavanja RTO naredbe

RTO naredba se razlikuje od TON naredbe samo po tome što se akumulirana vrednost ne resetuje, već zadržava i onda kada uslov postane neistint. Drugim rečima, ovaj tajmer počinje da radi kada uslov postane istinit, i nastavlja sa radom povećavajući akumuliranu vrednost sve dok je uslov istinit. Kada uslov postane neistinit, tajmer prekida rad, ali se akumulirana vrednost pri tome ne menja. To znači da će kada uslov ponovo postane istinit, tajmer nastaviti sa radom i prethodno izmerenom vremenu (ACC) dodavati nove vrednosti. Na taj način ovaj

tajmer omogućuje da se kumulativno mere intervali vremena u kojima je uslov bio istinit (Sl. 4-20).

Bitovi stanja RTO tajmera menjaju se u toku programskog sken ciklusa na sledeći način:

DN - Timer done bit se postavlja na 1 kada je ACC≥PRE (časovnik je izmerio zadato vreme). On se resetuje na 0 pomoću posebne RES naredbe.

EN - Timer enable bit se postavlja na 1 kada je uslov u rangu istinit (rad časovnika je omogućen) i resetuje na 0 kada uslov postane neistinit (rad časovnika je onemogućen).

TT - Timer timing bit se postavlja na 1 kada je uslov istinit i ako je ACC≤PRE (časovnik radi). On se resetuje na 0 kada uslov postane neistinit ili kada se DN bit postavi na 1 (časovnik prestaje sa radom).

- ***Reset naredba (RES)***

RES naredba je naredba akcije i koristi se za resetovanje tajmera. Kada je uslov istinit ova naredba se izvršava tako što se u tajmeru čija je adresa navedena u RES naredbi, resetuju na nulu bitovi DN, TT i EN, kao i akumulirana vrednost (ACC). S obzirom na način rada očigledno je da se RES naredbe ne sme koristiti za TOF tip časovnika.

4.8.2 Naredbe za prebrojavanje događaja

4.8.2.1 Parametri brojača

Svakom brojaču pridružena su sledeća dva parametra:

Zadata vrednost (*preset value* - PRE) je vrednost kojom se definiše ukupni broj događaja koje brojač treba da registruje pre nego što se generiše signal koji označava da je brojač završio rad. Zadata vrednost za brojač može da se kreće u intervalu od 0 do +32767.

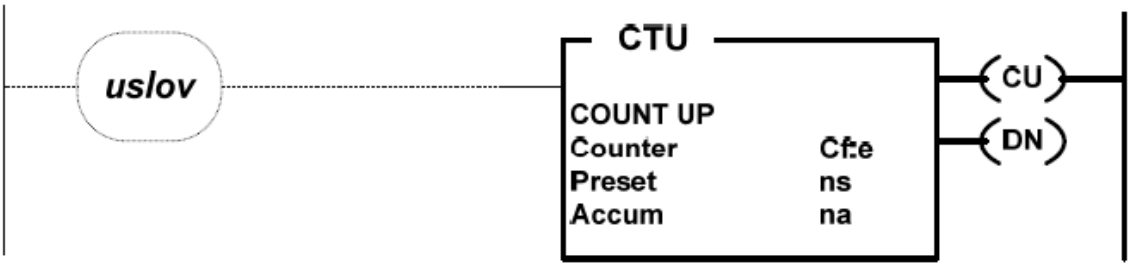
Akumulirana vrednost (***accumulated value* - ACC**) predstavlja broj događaja koje je brojač registrovao u nekom trenutku.

4.8.2.2 Naredbe brojača

Postoje dva osnovna tipa brojača *brojač unapred* (*CTU – count up*) i *brojač unazad* (*CTD – count down*). Obe naredbe su naredbe *akcije*, što znači da se smeštaju u desni deo ranga. Oba brojača broje promene 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 zadate vrednosti se signalizira postavljavanjem odgovarajućeg bita – *done bit* (DN) – na 1, ali se brojanje i dalje nastavlja. Prebrojani iznos se može izbrisati jedino posebnom *RES* naredbom.

Jedina razlika između dva tipa brojača sastoji se u tome što prvi (CTU) broji unapred od 0 do 32767, i postavlja *overflow bit* (OV) na 1 kad pređe 32767, dok drugi (CTD) broji unazad, od 0 do –32768, i postavlja *underflow bit* (UN) kad pređe –32768.

- ***Count up (CTU)***



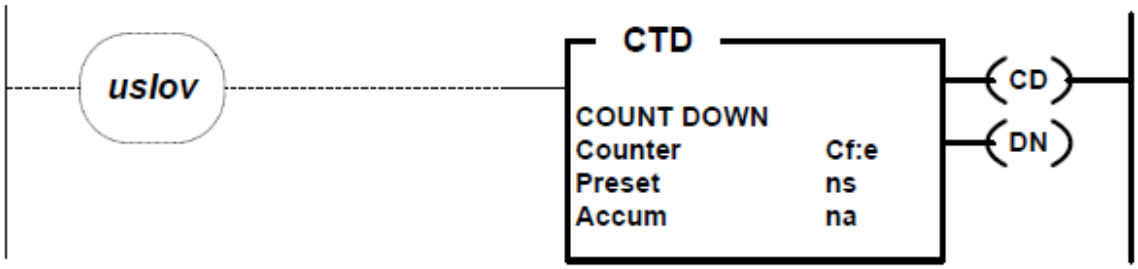
Bitovi stanja brojača menjaju se u toku programskog sken ciklusa na sledeći način:

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 reči važi: 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 reči, važi: –32768-1= 32767), i nastavlja da broji unazad od te vrednosti.

DN - done bit se postavlja na 1 kada je ACC≤PRE;

CD - Count down 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.