Leder programiranje

1 Rang

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 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. 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*.

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. 1-1).

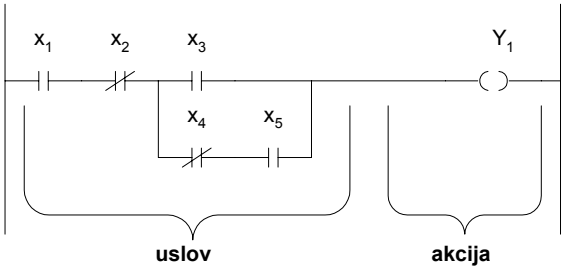
- ***Uslov*** – U osnovi, grafički simboli na levoj strani ranga odnose se ili na stanja signala koji predstavljaju fizičke ulaze PLC-a, i čije su vrednosti tokom ulaznog dela sken ciklusa smeštene u *input image file*, ili na stanja internih promenljivih, čije su vrednosti smeštene u odgovarajućim datotekama. 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 istinit, 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.

- ***Akcija*** – Grafički simboli na desnoj strani ranga odnose se ili na fizički izlaz (promenljive smeštene u *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 desnoj 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 u nizu kako su oni definisani. U svakom rangu ispituje se istinitost uslova i ukoliko je uslov istinit izvršavaju se odgovarajuće naredbe u desnom delu ranga. To znači da promenljive na 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, međutim, 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.



Sl. 1-1 - Leder rang

2 Bit naredbe

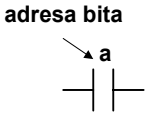
Bit naredbe su, kao što samo ime kaže naredbe čiji su operandi *bitovi*. Sa gledišta lokacije operanada, to znači da se oni najčešće nalaze u datoteci 3 (bit file), digitalnim ulaznim ili izlaznim datotekama (input image file 1 ili output image file 0) ili u korisničkim datotekama bit tipa. Pored toga, 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*).

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)**

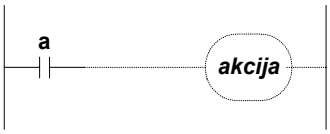
Grafički simbol



Tablica istinitosti

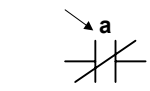
Stanje bita "a"	Vrednost XIC naredbe
0	False
1	True

Položaj u rangu



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

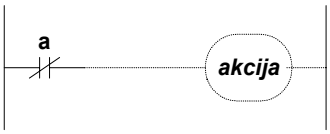
Grafički simbol



Tablica istinitosti

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

Položaj u rangu



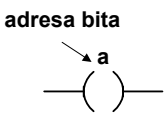
Nazivi ove dve naredbe potiču od ispitivanja binarnih signala koji dolaze sa prekidačkih kola. U tom smislu *XIC* naredba se odnosi na *normalno otvoren prekidač*, dok se *XIO* naredba odnosi na *normalno zatvoren prekidač*.

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)**

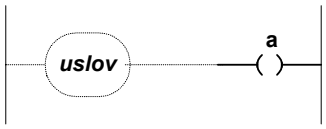
Grafički simbol



Akcija

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

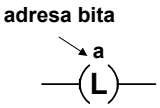
Položaj u rangu



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)**

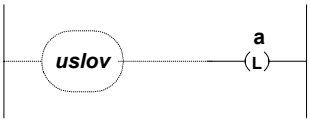
Grafički simbol



Akcija

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

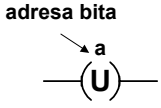
Položaj u rangu



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)**

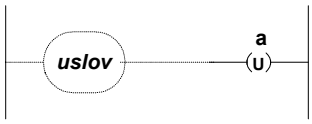
Grafički simbol



Akcija

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

Položaj u rangu

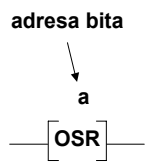


OTU naredbom se adresovani bit može isključivo postaviti na 0. Pri tome, vrednost bita se postavlja (lečuje) na 0 u prvom skenu u kome je *uslov* ispunjen. Nakon toga ova naredba postaje neosetljiva na vrednost *uslova*.

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

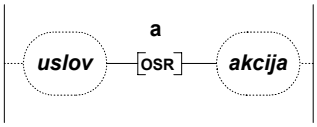
2.3 Bit triger naredba

- **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)



OSR naredba omogućava da se obezbedi izvodjenje neke akcije *samo jedanput*. Potrebno je da se istakne da je ovo specifična naredba koja 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".

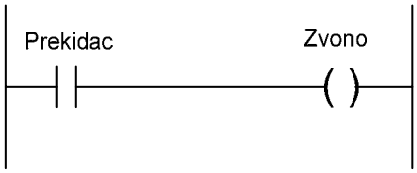
Potrebno je istaći da bit čija je adresa pridružena ovoj naredbi ne predstavlja vrednost naredbe. Naime, ovaj bit se koristi kao interna promenljiva i služi za pamćenje *vrednosti uslova* koji prethodi *OSR* naredbi. Vrednost ovog bita je 1 ako je *uslov istinit*, odnosno 0 ako je *uslov neistinit*. 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.

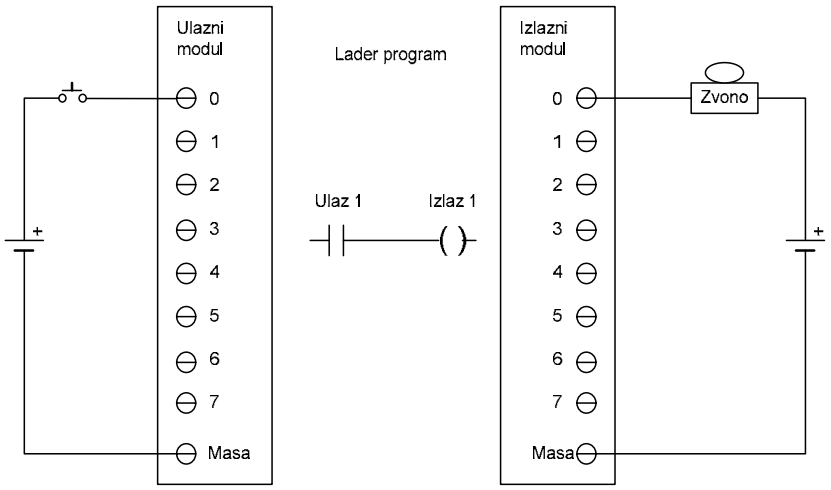
2.4 Kontakti

Većina ulaza u PLC su jednostavni uređaji koji mogu biti u stanju uključeno (*on*) ili u stanju isključeno (*off*). Ovakvi ulazi su prekidači i digitalni senzori koji detektuju uslove tipa: objekat je prisutan, puno/prazno i td. U leder programu stanje prekidača se ispituje naredbama XIC, za normalno otvorene prekidače/senzore, i XOC, za normalno zatvorene prekidače/senzore.



Sl. 2-1 Jenostavan leder dijagram.

Prekidač za zvono na ulaznim vratima je primer normalno otvorenog prekidača. Pritiskom na prekidač, zvono se spaja sa izvorom napajanja, struje počinje da teče i zvono zvoni. (Ukoliko bi smo za ovu namenu koristili normalno zatvoren prekidač, zvono bi zvonilo za sve vreme dok je prekidač nepritisnut, a ne bi zvonilo samo dok je prekidač pritisnut, što je očigledno neželjeno ponašanje.) Odgovarajući leder dijagram prikazan je na Sl. 2-1. Program se sastoji iz samo jednog ranga, koji u delu uslova sadrži XIC naredbu, koja predstavlja prekidač, a u delu akcija OTE naredbu, koja predstavlja zvono. Ako je uslov tačan (prekidač pritisnut), akcija se izvršava (zvono se pobuđuje). Konceptualni prikaz PLC sistema za ovu namenu dat je na Sl. 2-2 (iako je sasvim jasno da za ovu namenu PLC predstavlja krajnje neracionalno rešenje). Prekidač je preko eksternog izvora napajanja priključen na ulaz 1, dok je zvono, takođe preko eksternog izvora napajanja, priključeno na izlaz 1 PLC kontrolera. Centralni deo slike prikazuje logiku po kojoj procesor određuje izlaz u zavisnosti od ulaza.



Sl. 2-2 Konceptualni pogled na PLC sistem.

Zamislimo senzor koji treba da detektuje prisustvo metalnog predmeta na pokretnoj traci. I za ovu namenu, senzor sa normalno otvorenim kontaktima predstavlja logičan izbor – senzor se uključuje kada metalni predmet dođe ispred senzora; kada metalni prođe, senzor ponovo prelazi u isključeno stanje.

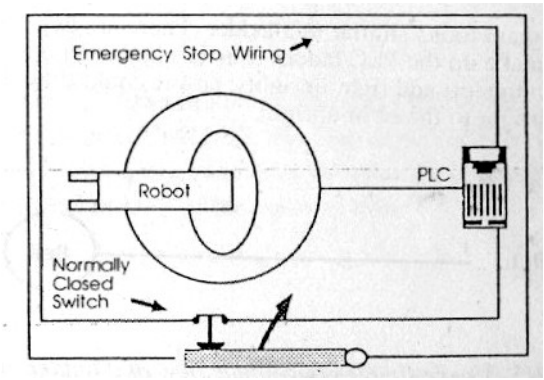
Normalno zatvoreni prekidači/senzori se koriste kada treba obezbediti veću sigurnost sistema. Ovakav prekidač je u stanju zatvoreno (propušta struju) za sve vreme dok nije pritisnut, dok se pritiskom na prekidač njegovi kontakti otvaraju (struja ne teče). Alarmni sistem je primer sistema gde je poželjno koristiti normalno zatvorene prekidače. Pretpostavimo da alarmni sistem treba da detektuje otvaranje ulaznih vrata. Ova jednostavna funkcija se može ostvariti pomoću normalno-otvorenog prekidača (slučno kao u primeru zvona na ulaznim vratima): kada se vrata otvore, prekidač se zatvara i alarm se uključuje. Međutim, rešenje sa normalno-otvorenim prekidačem ima jedan ozbiljan nedostatak. Pretpostavimo da se prekidač pokvario ili da se žica kojom je prekidač povezan sa PLC modulom prekinula. Očito, u tom slučaju, alarm se nikada neće uključiti, bez obzira da li su ulazna vrata otvorena ili ne. Drugim rečima, vlasnik kuće nije dobio informaciju da se sistem pokvario i zato nastavlja da koristi sistem kao da je sve u redu.

Ispravno rešenje je ono koje može da obezbedi aktiviranje alarma kada se vrata otvore, ali i onda kada sistem otkaže. Bolja varijanta je da se alarm aktivira zato što je sistem otkazao, iako nema provalnika, nego da je provala u toku, a alarm "ćuti" zato što je prekidač pokvaren. Ovakvo ponašanje se može lako realizovati uz pomoć normalno-zatvorenog

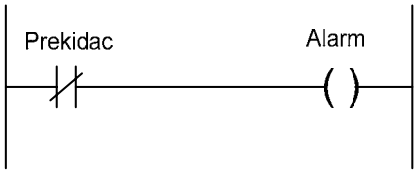
prekidača – otvaranje vrata i prekid žice (slučajan ili nameran) ima isti efekat: prekid strujnog kola.

Slična razmatranja imaju čak i veći značaj kada se radi o industrijskim primenama, gde otkaz neke mašine može uzrokovati veliku štetu ili povrede ljudi. Zato se prilikom projektovanja sistema i razvoja leder programa posebna pažnja posvećuje bezbednosti sa ciljem da u slučaju otkaza sistem bude postavljen u stanje koje će biti bezbedno za ljude i sam proces.

Razmotrimo sistem sa (Sl. 2-3). Slika predstavlja proizvodnu ćeliju u kojoj radi robot. Ćelija je ograđena ogradom sa jednim ulaznim vratima. Kao kontroler ćelije koristi se PLC. Da bi se osiguralo da niko ne može ući u ćeliju dok robot radi, iskorićen je sigurnosni prekidač. Ako neko uđe u ćeliju, PLC će detektovati da je prekidač otvoren i uključiće alarm. Za ovu namenu treba koristiti normalno-zatvoren prekidač. Ako se žica koja povezuje prekidač sa PLC kontrolerom prekine, PLC će "misliti" da je neko ušao u ćeliju i aktiviraće alarm. Kaže se da je ovako projektovan sistem *bezbedan na otkaze*.



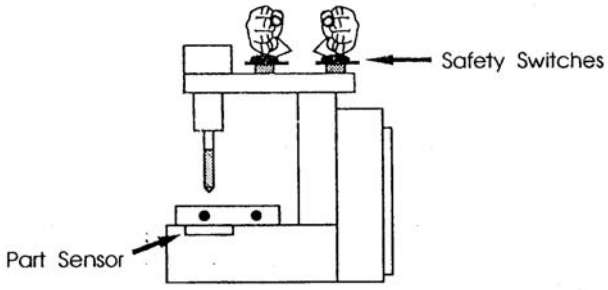
Sl. 2-3 Obezbeđenje proizvodne ćelije. (*Normally Closed Switch – Normalno zatvoren prekidač*)



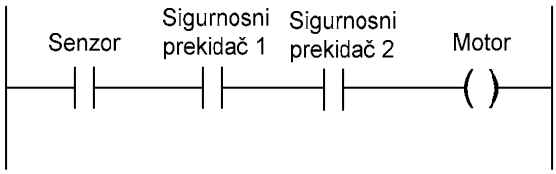
Sl. 2-4 Rang leder dijagrama sa normalno-zatvorenim prekidačem.

Na Sl. 2-4 je prikazan leder program za prethodno opisanu primenu. Leder program se sastoji iz samo jednog ranga koji u delu uslova sadrži XCI naredbu, koja predstavlja normalno-zatvoreni prekidač, a u delu akcija OTE naredbu, koja predstavlja alarm. Uslov će biti tačan i akcija će biti izvršena, ako je na odgovarajućem ulazu PLC modula prisutna 0, odnosno ako je prekidač otvoren, tj. pritisnut.

U istom rangu može se naći više od jednog prekidača. Na primer, zamislimo mašinu za bušenje rupa. Motor bušilice se uključuje pod uslovom da je predmet koji se buši prisutan i da je operater pritisnuo oba sigurnosna prekidača (Sl. 2-5). Na Sl. 2-6 je prikazan odgovarajući leder program. Program se sastoji iz samo jednog ranga koji u delu uslova sadrži serijsku vezu tri XIO naredbi od kojih prva odgovara senzoru za detekciju prisustva predmeta, dok druge dve odgovaraju sigurnosnim prekidačima. Seriska veza prekidača realizuje logički AND uslov (da bi se akcija obavila, svi prekidači moraju biti uključeni).

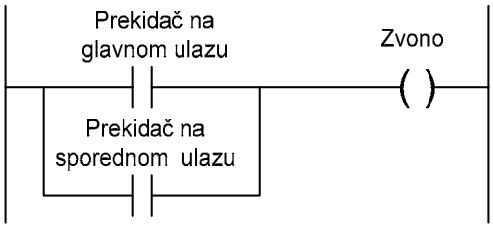


Sl. 2-5 Bušilica sa sigurnosnim prekidačima.



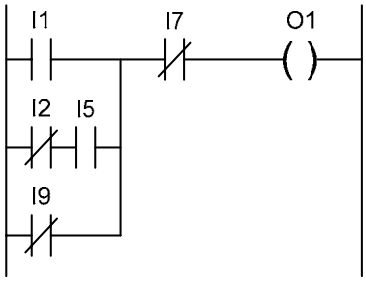
Sl. 2-6 Rang leder dijagrama sa serijskom vezom prekidača.

Često se javlja potreba da se izlaz aktivira ako je ispunjen barem jedan od više uslova. Zamislimo zgradu sa glavnim i sporednim ulazom. Na oba ulaza postoje prekidači za zvono. Pritisak na bilo koji od ova dva prekidača uključuje zvono. Na Sl. 2-7 je prikazan odgovarajući leder program. Leder program se sastoji iz samo jedan rang, koji u delu uslova sadrži grananje, tj. dve paralelne putanje (ili uslova) koje mogu uključiti zvono. Grananje predstavlja logičku OR operaciju nezavisnih uslova. Zvono zvoni ako je pritisnut prekidač na glavnom ulazu ili prekidač na sporednom ulazu ili oba prekidača istovremeno.



Sl. 2-7 Rang leder dijagrama sa paralelnom vezom prekidača.

U opštem slučaju, rang leder dijagrama može sadržati proizvoljnu kombinaciju redno i paralelno vezanih prekidača (Sl. 2-8).

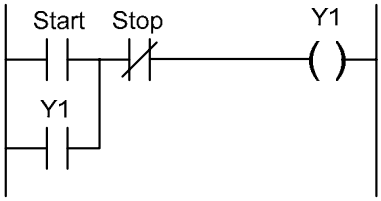


Sl. 2-8 Rang sa redno-paralelnom vezom prekidača.

2.5 Start/stop kolo

Start/stop kolo se veoma često koristi u industrijskim primenama. Na primer, mašina može imati start prekidač za početak rada i stop prekidač za zaustavljanje. Primer leder dijagrama start/stop kola prikazan je na Sl. 2-9. Start je normalno-otvoren, a Stop normalno-zatvoren prekidač. Pritiskom na prekidač Start, uslov ranga postaje tačan, a izlaza Y1 se aktivira. Uočimo da se izlaz Y1 koristi i kao ulaz. S obzirom da je sada Y1=1, uslov ostaje

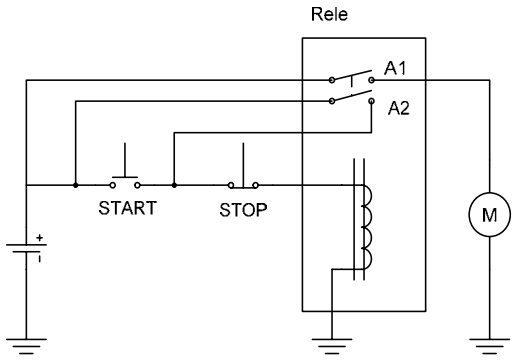
tačan, a izlaz aktivan i nakon otpuštanja prekidača Start. Pritiskom na prekidač Stop, uslov ranga postaje netačan, a izlaz se deaktivira. Pošto je sada Y1=0, uslov ostaje netačan, a izlaz neaktivan i nakon otpuštanja prekidača Stop. Da bi se izlaz ponovo aktivirao potrebno je ponovo pritisnuti taster Start. Opisani postupak formiranja start/stop kola se zove *samodržanje*, s obzirom da izlazna promenljiva zadržava vrednost i posle prestanka uslova za njeno aktiviranje.



Sl. 2-9 Start/stop kolo.

Ponašanje Start/stop kola je identično ponašanju SR leča, pri čemu prekidač Start ima ulogu setovanja, a Stop ulogu resetovanja leča.

Osim u leder programima, Start/stop kolo se često realizuje i u relejnoj tehnici, uz pomoć relea sa dva normalno-otvorena kontakta. Na Sl. 2-10 je prikazana realizacija start/stop kola za upravljanje motorom. Jedan kontakt relea, A1, kontroliše napajanje motora, dok se drugi kontakt, A2, koristi za realizaciju start/stop kola. Pritiskom na prekidač Start, kroz namotaj relea počinje da teče struja, oba kontakta, A1 i A2 se zatvaraju i motor počinje da radi. Ovakvo stanje se zadržava i nakon otpuštanja prekidača Start, jer struja za namotaj relea nastavlja da teče kroz zatvoren kontakt A2. Pritiskom na prekidač Stop, struja kroz namotaje relea se prekida, kontakti A1 i A2 se otvaraju, motor prestaje da radi a kroz namotaje relea više ne protiče struja.

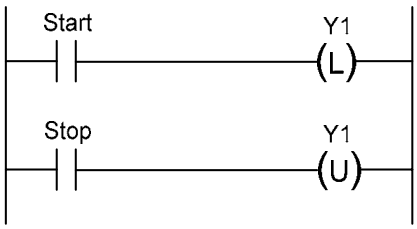


Sl. 2-10 Realizacija Start/Stop kola pomoću relea.

Na prvi pogled čini se da start/stop kolo predstavlja neracionalno rešenje, jer se ista funkcija (prosto aktiviranje/deaktiviranje izlaza) može ostvariti pomoću samo jednog, dvo-položajnog prekidača, koji bi direktno upravljao izlazom. Razlog za korišćenje start/stop kola je bezbednost. Pretpostavimo da u sistemu sa Sl. 2-10, motor radi (kroz namotaje relea teče struja) i da u jednom trenutku dođe do nestanka električne energije. Motor se zaustavlja, a kontakti relea otvoraju. Kada naknadno eletrična energija dođe, motor ostaje isključen, a da bi se ponovo uključio neophodno je pritisnuti prekidač Start. Ako bi smo za upravljanje motorom koristili samo jedan prekidač koji bi direktno kontrolisao napajanje motora, motor bi po dolasku električne energije nastavio da radi, zato što je prekidač ostao uključen, što može biti kritično sa stanovišta bezbednosti.

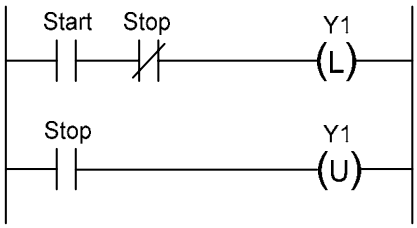
Start/stop kolo se može realizovati i uz pomoć naredbi OTL i OTU, kao što je prikazano na slici. Uočimo da su sada oba ulaza, Start i Stop, sa normalno-otvorenim kontaktima i da se

obe naredbe OTL i OTU odnose na isti izlaz, Y1. Postavljanjem ulaza Start na 1, izvršava se naredba OTL, koja postavlja izlaz Y1 na 1 (Y1 se setuje), koji ostaje 1 i kada se Start vrati na 0. Da bi se izlaz Y1 postavio na 0 (tj. resetovao), potrebno je postaviti Stop=1, što aktivira naredbu OTU. Izlaz Y1 zadržava vrednost 0 i nakon postavljanja ulaza Stop na 0.



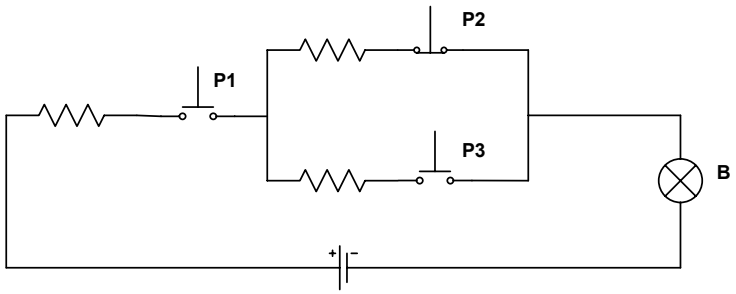
Sl. 2-11 Realizacija start/stop kola pomoću naredbi OTL i OTU.

U rešenju sa Sl. 2-11, kritična situacija je ona kada su oba prekidača, Start i Stop, zatvorena (=1). Naime, ovakva situacija nesme da se javi na ulazu, jer pod tim uslovom kolo može početi da osciluje. Da bi se predupredilo ovakvo neželjeno ponašanje, leder program sa Sl. 2-11 može se proširiti XIC naredbom u prvom rangu koja će sprečiti da pri Start=1 uslov postane tačan ako je Stop=1 (Sl. 2-12). Očigledno, u rešenju sa Sl. 2-12 ulaz Stop ima viši prioritet, tako da se pri Start=Stop=1 kolo resetuje.



Sl. 2-12 Start/stop kolo koje rešava problem Start=Stop=1.

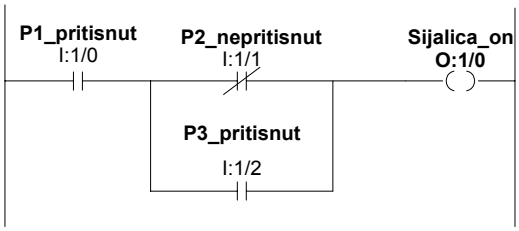
Primer 1: Napisati leder program za PLC koji zamenjuje električno kolo prikazano na Sl. 2-13, u kome se pomoću prekidača vrši paljenje i gašenje sijalice. Kao što se vidi, sijalica S će svetleti kada je zatvoren prekidač P_1 i jedan od prekidača P_2 ili P_3.



Sl. 2-13 Elektročno kolo.

Rešenje: Za formiranje programa neophodno je raspolagati informacijom o vrsti prekidača kao i načinu njihovog vezivanja za U/I modul. Neka su P_1, P_2 i P_3 tasteri, pri čemu su P_1 i P_3 u normalnom stanju otvoreni, a P_2 je u normalnom stanju zatvoren. To znači da će pritisak na P_1 ili P_3 dati digitalni signal koji predstavlja logičku jedinicu, dok će pritisak na P_2 dati logičku nulu. Predpostavimo da su prekidači P_1, P_2 i P_3 kao i sijalica S prikačeni na kombinovani digitalni U/I modul koji je smešten u slot 1 PLC-a. Pretpostavimo nadalje da se linije sa prekidačkih kola dovode na pinove 0, 1 i 2 ulaznog dela modula, dok je kolo u kome se nalazi sijalica vezano za pin 0 izlaznog dela modula. Shodno tome, adrese prekidača P_1, P_2 i P_3 su respektivno I:1/0, I:1/1 i I:1/2, dok je adresa sijalice O:1/0.

U cilju formiranja levog dela ranga treba uočiti da je *uslov* za paljenje sijalice da se istovremeno pritisne taster P_1 i jedan od tastera P_2 ili P_3. Budući da su tasteri P_1 i P_3 normalno otvoreni, pritisak na njih dovodi do zatvaranja odgovarajućih prekidačkih kola, tako da se može detektovati pomoću *XIC* naredbe, koja će dobiti vrednost *istinit* kada su vrednosti odgovarajućih bitova u slici ulaza postavljene na 1. Pritisak na taster P_2 koji je normalno zatvoren, dovodi do otvaranja njegovog prekidačkog kola, što znači da se može detektovati pomoću *XIO* naredbe, koja će dobiti vrednost *istinit* kada je vrednost odgovarajućeg bita u slici ulaza postavljena na 0. Konačno, kako se nad tasterima 2 i 3 zahteva logička *ILI* opreracija, to odgovarajuće naredbe moraju biti vezane paralelno. Ispunjenost *uslova* treba da obezbedi da se na izlaznom pinu generiše naponski signal koji će da prouzrokuje paljenje sijalice. Ovaj zahtev se može ostvariti *OTE* naredbom. U skladu sa time odgovarajući rang leder programa ima izgled kao na Sl. 2-14.

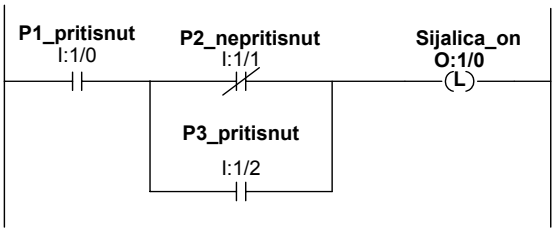


Sl. 2-14 Rang leder programa koji realizuje električno kolo.

Svakom bitu koji se koristi u leder programu može se pridružiti simboličko ime. U principu nema nikakvih posebnih pravila u pogledu davanja imena. Ipak praksa je pokazala da je pogodno da se ime formira tako da što vernije opisuje fizičko značenje signala na koji se odnosi. Pored toga, u cilju lakše provere ispravnosti programa, pogodno je da se ime formira tako da odgovara stanju pri kome bit ima vrednost 1. Poštujući taj princip, u ovom primeru je bitu I:1/0 dato ime *p1_pritisnut*, dok je bitu I:1/1 dato ime *p2_nepritisnut*. U skladu sa time prva *XIC* naredba koja ispituje vrednost bita na adresi I:1/0 dobiće vrednost *istinit* ako bit ima vrednost 1 što znači da taster **P_1 jeste pritisnut**. Isto tako *XIO* naredba koja ispituje vrednost bita na adresi I:1/1 dobiće vrednost *istinit* ukoliko bit nema vrednost 1, što znači da taster P_2 **nije nepritisnut**.

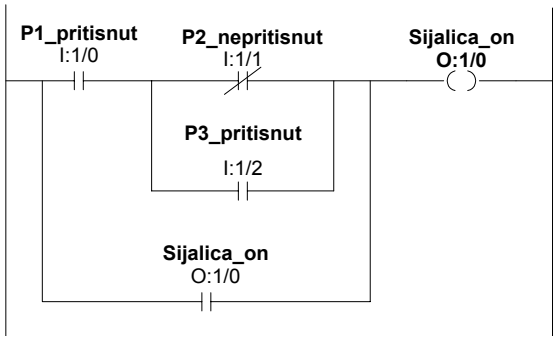
Opisani sistem će ispravno raditi samo dotle dok se odgovarajući tasteri drže pritisnuti. Naime, čim se taster otpusti on se vraća u normalni položaj i u sledećem sken ciklusu, uslov više neće biti ispunjen, pa će se sijalica ugasiti. Prirodno je međutim da se zahtev postavi tako da sijalica nastavi da svetli i posle otpuštanja tastera.

Postavljeni zadatak može se rešiti tako što će se na neki način upamtiti da je uslov za paljenje sijalice u nekom trenutku bio ispunjen. U tu svrhu može se na izlaznom delu ranga umesto *OTE* naredbe postaviti *OTL* naredba koja će obezbediti trajno postavljanje (lečovanje) izlaza (Sl. 2-15).



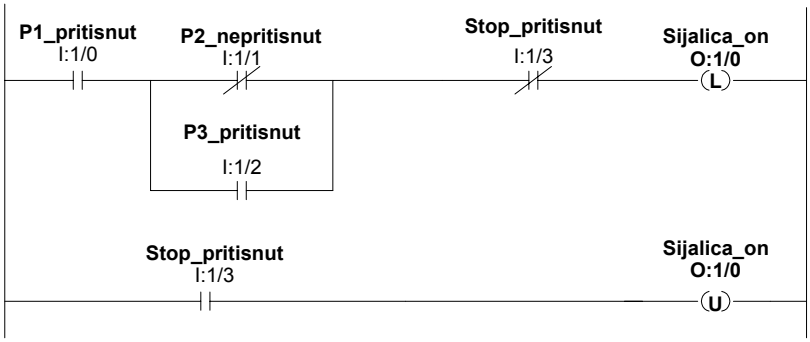
Sl. 2-15 – Trajno postavljanje izlaznog bita

Umesto korišćenjem naredbe *OTL* isti efekat se može postići i postupkom samodržanja. Naime, ako se u delu ranga koji predstavlja *uslov* doda još jedna paralelna grana sa *XIC* naredbom u kojoj se ispituje upravo bit koji se postavlja kao izlaz tog ranga (Sl. 2-16) onda će, čim se pritiskanjem tastera *uslov* prvi put ispuni, odgovarajući bit biti postavljen na 1, što znači da će u sledećim sken ciklusima naredba u paralelnoj grani stalno imati vrednost *istinit*, pa se vrednost izlaznog bita neće menjati sa promenom stanja tastera.



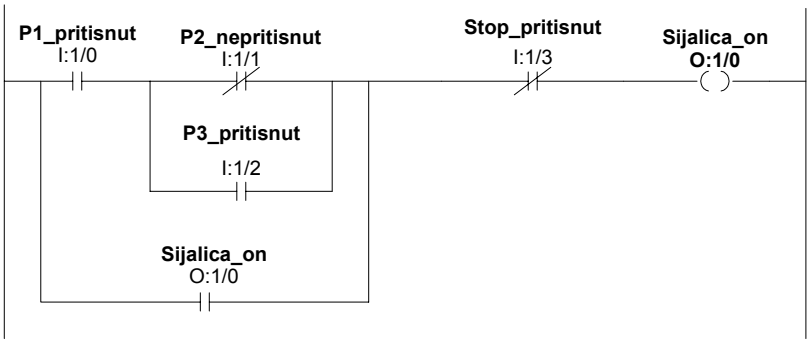
Sl. 2-16 – Postupak samodržanja

Potrebno je istaći da su poslednja dva primera formirana tako da će sijalica kad se jedanput upali nastaviti da svetli neograničeno dugo.



Sl. 2-17 – Paljenje i gašenje sijalice pomoću naredbi za lečovanje

Da bi se omogućilo i gašenje sijalice neophodno je da se sistemu doda još jedan taster (*Stop*). Pritiskom na ovaj taster, jedanput upaljena sijalica, bi bila isključena. Ako se pretpostavi da je ovaj taster normalno otvoren i da je vezan na pin 3 istog ulaznog modula, onda se postavljeni zadatak može realizovati na način koji je prikazan na slikama Sl. 2-18 i Sl. 2-19. Ovde je na red sa delom ranga kojim se ostvaruje paljenje sijalice, vezan uslov kojim se proverava da li Stop taster *nije pritisnu*. Sve dok Stop taster nije pritisnut, vrednost tog dela uslova je *istinita*, dakle on ne utiče na ponašanje sijalice. Kad se Stop taster pritisne, vrednost tog dela uslova postaje *neistinita*, a budući da je to redni (serijski) uslov, i vrednost celog ranga postaje *neistinita*, pa se aktivira se naredba *unletch* (Sl. 2-17), odnosno izlazni bti se *OTE naredbom* postavlja na 0 (Sl. 2-18).



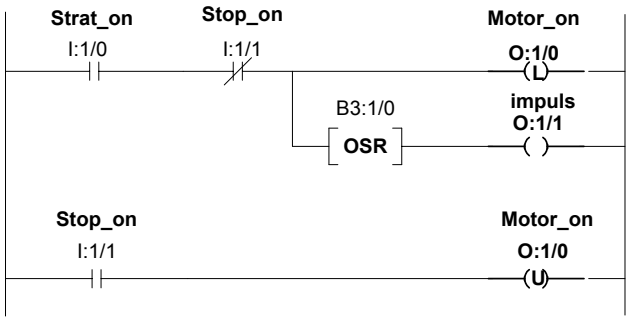
Sl. 2-18 – Paljenje i gašenje sijalice pomoću postupka samodržanja

Naravno, ceo problem oko paljenja i gašenja sijalice bio bi rešen rangom koji je prikazan na Sl. 2-14, da su umesto tastera korišćeni dvopoložajni prekidači.

Primer 2: Posmatra se sistem koji počinje da radi kada se pritisne START taster. Sistem nastavlja sa radom sve dok se ne pritisne stop taster. Istovremeno se zahteva da se prilikom započinjanja rada sistema generiše impulsni signal koji realizuje brzo "zamrzavanje" nekog LCD displeja. Potrebno je da se formira leder program koji će podržavati rad opisanog sistema uz predpostavku da se on pušta u rad pomoću jednog "on/off" motora.

Neka su kola koja sadrže START i STOP taster kao i kolo za pobudu motora vezani za U/I modul koji je smešten u slotu 1 PLC-a i to na pinove kojima respektivno odgovaraju adrese I:1/0, I:1/1 i O:1/0. Neka je nadalje signal koji upravlja zamrzavanjem displeja vezan za izlazni pin čija je adresa O:1/1. Odgovarajući leder program dat je na Sl. 2-19.

U prvom rangu *uslov* je ispunjen ako je START taster pritisnut i STOP taster nije pritisnut. U tom slučaju generisaće se signal za start motora i on će biti lečovan, tako da se njegova vrednost neće menjati ukoliko se zbog otpuštanja START tastera promeni vrednost *uslova*. Istovremeno će, prilikom pritiska START tastera, *OSR* naredba detektovati promenu *neistinit/istinit* što će dovesti do toga da ona u tom sken ciklusu dobije vrednost *istinit* tako da će se izvršiti *OTE* naredba kojom se na izlazu O:1/1 generiše potreban impulsni signal. Sve dok se ne pritisne STOP taster, vrednost uslova u drugom rangu biće *neistinita*, tako da se naredba *unletch* neće izvršavati. Kada se pritisne STOP taster, uslov u prvom rangu postaje *neistinit*, što znači da se *letch* naredba ne izvršava. Međutim, vrednost uslova u drugom rangu postaje *istinita*, pa se izvršava *unletch* naredba, i time signal za start motora dobija vrednost "logičke" nule, pa se motor zaustavlja.



Sl. 2-19 – Start/Stop leder program

Primer: Realizacija sekvence

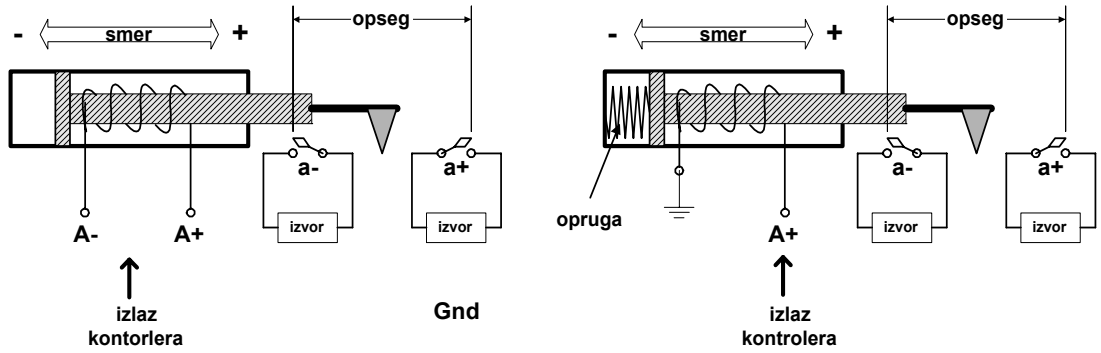
Zadatak: Dat je sistem koji sadrži jedan jednosmerni solenoid (A) i dva dvosmerna (B i C). Potrebno je realizovati sledeću sekvencu pomeranja klipova: A+ B+ C+ B- A- C- . Pri tome se predpostavlja da su u početnom trenutku svi klipovi uvučeni. Granični prekidači koji indiciraju uvučenost klipa A i B su *normalno zatvoreni*, dok su svi ostali granični prekidači *normalno otvoreni*. Sistem se pušta u rad pomoću pritiska na taster i prestaje sa radom kada se jedanput izvrši zahtevana sekvenca.

Solenoid

Solenoid je elektromehanički aktuator, čijim radom se upravlja pomoću elektromagnetne sile proizvedene u namotaju. U principu, rad solenoida zasniva se na struji koja postoji u namotaju i koja proizvodi magnetno polje. U zavisnosti od smera struje, menja se i smer sile magnetnog polja koja privlači gvozdeno jezgro ka centru namotaja ili ga odbija od centra. Postoje dva tipa solenoidnih aktuatora:

- *Jednosmerni* solenoid – kod koga postoji samo jedan izvod za napajanje, tako da struja ima uvek isti smer, što znači da se i jezgro pod dejstvom magnetne sile može pomerati samo u jednom smeru. U odsustvu napajanja solenoida, mehanička opruga vraća jezgro u početni položaj.
- *Dvosmerni solenoid* – kod koga postoje 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, uz uslov da ne postoji neka mehanička sila (npr. sila zemljine teže, ako je solenoid u vertikalnom položaju) koja bi izazvala njegovo kretanje. Drugim rečima, u odsustvu napajanja, solenoid se nalazi u slobodnom stanju.

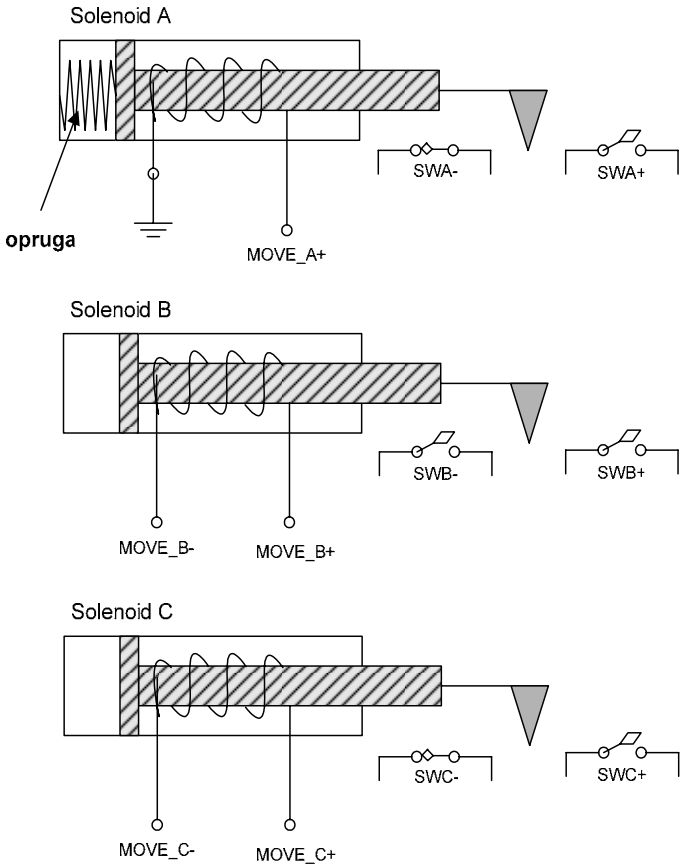
  Šematski dijagram oba tipa solenoida prikazan je na Sl. 2-20. Potrebno je da se istakne da solenoid po pravilu ima i dva granična prekidača koji omogučavaju da se detektuje kada jezgro dođe u krajnji desni ili krajnji levi položaj.



Sl. 2-20 Dvosmerni i jednosmerni solenoid.

Rešenje: Tri solenoida, A, B i C, prikazana su na Sl. 2-21. A je jednosmerni solenoid i zbog toga ima samo jedan ulaz, označen kao MOVE_A+, koji kada je pobuđen izvlači jezgro, a po prestanku pobude jezgro se pod dejstvom opruge vraća u polazni, uvučeni položaj. Preostala dva solenoida B i C su dvosmerni i zato imaju po dva ulazna priključka. Ako se napajanje

dovede na priključak MOVE_B+ (MOVE_C+) jezgro solenoida se izvlači; ako se napajanje dovede na priključak MOVE_B- (MOVE_C-) jezgro solenoida se uvlači. Svaki solenoid ima dva granična prekidača koji se aktiviraju kada jezgro dođe u krajnji uvučeni položaj (SWA-, SWB- i SWC-), odnosno u krajnji izvučeni položaj (SWA+, SWB+ i SWC+). Dva od ukupno šest graničnih prekidača, SWA- i SWC- imaju normalno-zatvorene (NC) kontakte, dok preostali imaju normalno-otvorene kontakte (NO).

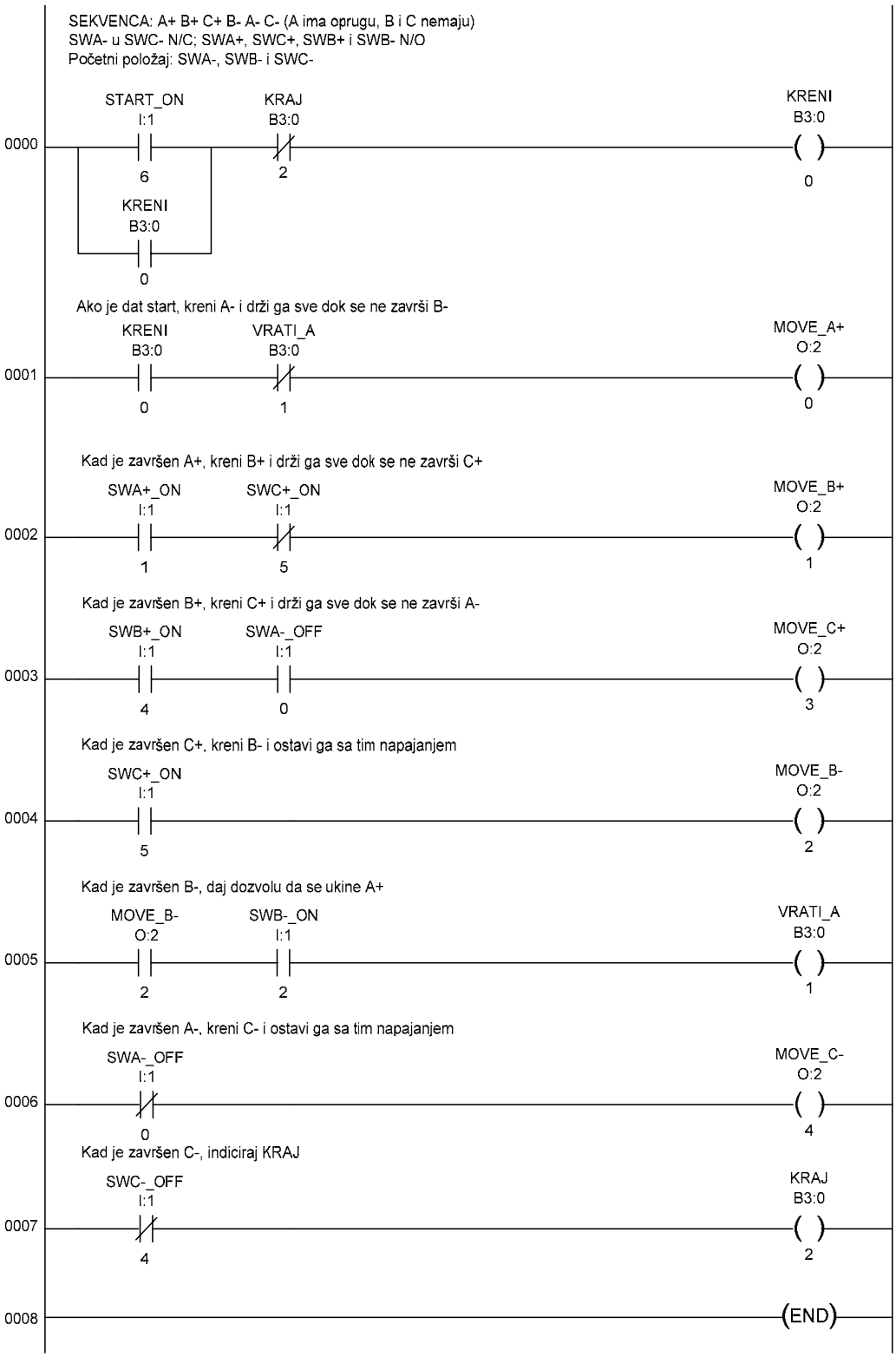


Sl. 2-21 Tri solenoida

Granični prekidači se povezuju na pinove ulaznog, a ulazni priključi solenoida na pinove izlaznog PLC modula. Usvojićemo sledeće adrese:

Ulaz	Adresa	Izlaz	Adresa
SWA-	I:1/0	MOVE_A+	O:2/0
SWA+	I:1/1	MOVE_B+	O:2/1
SWB-	I:1/2	MOVE_B-	O:2/2
SWB+	I:1/3	MOVE_C+	O:2/3
SWC-	I:1/4	MOVE_C-	O:2/4
SWC+	I:1/5		
START	I:1/6		

Jedno moguća rešenje postavljenog zadatka dato je na Sl. 2-22. Potrebno je da se obrati pažnja na činjenicu da se, da ne bi došlo do slučajnog pomeranja klipova (izazvanog recimo nekim opterećenjem), oni u izvučenom položaju drže pod naponom, sve dok ne dođe trenutak za njihovo uvlačenje.



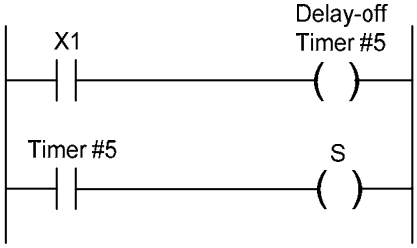
Sl. 2-22 Realizacija sekvence.

3 Naredbe za merenje vremena i prebrojavanje događaja – časovnici 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*** (***timer***). U suštini časovnik izražava vreme kao umnožak 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.

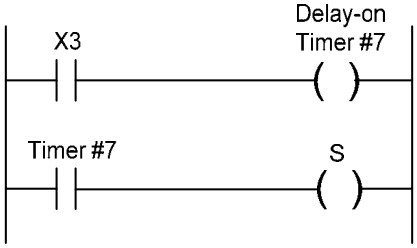
3.1 Realizacija časovnika

Zamislimo sijalicu u ulazu neke zgrade. Sijalica se pali prekidačem (tasterom) i ostaje upaljena neko zadato, fiksno vreme (npr. 2 min.) dovoljno da se prođe kroz ulaz i uđe u stan, a zatim se "sama" gasi. Dakle, izlaz (sijalica) se uključuje istog trenutk kada se aktivira ulaz (taster); časovnik odbrojava vremenske intervale i kada dostigne zadatu vrednost isključuje izlaz (sijalicu). Ovo je primer takozvanog *delay-off* časovnika.



Sl. 3-1 Delay-off časovnik.

Razmotrimo Sl. 3-1. Kada se prekidač X1 uključi, časovnik, označen kao Delay-off Timer #5, se uključuje i merenje vremena započinje. Časovnik se koristi u drugom rangu za definisanje uslova. Ako je časovnik aktivan (on), sijalica S je uključena. Nakon što je zadato vreme isteklo (u ovom primeru, 2 min.), časovnik se isključuje, uslov u drugom rangu postaje netačan i sijalica se isključuje.



Sl. 3-2 Delay-on časovnik.

Drugi tip časovnika zove se *delay-on* časovnik (Sl. 3-2). Aktiviranje ulaza X3 startuje delay-on časovnik, označenog kao Delay-on Timer #7. Časovnik započine brojanje, ali ostaje u stanju 0 (off) sve do isteka zadatog vremena. Po isteku zadatog vremena stanje časovnika postaje 1 (on) i sijalica S iz drugog ranga se uključuje.

U seriji kontrolera SLC 5 časovnici i brojači su realizovani softverski, i koriste se kao naredbe *akcije*. Kao što je već istaknuto ne postoji nikakvo posebno ograničenje u pogledu njihovog broja.

Pri korišćenju časovnika i brojača neophodno je da se definišu sledeći parametri.

- **Vremenska baza (*time base*)** određuje dužinu *osnovnog intervala vremena*. Kod fiksnog kontrolera SLC/500 i modularnog SLC 5/01, vremenska baza je definisana kao 0.01 sec. Kod kontrolera SLC 5/02, 5/03 i 5/04 bira se jedna od dve moguće vrednosti: 0.01 sec ili 1.0 sec.
- **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), odnosno ukupni broj događaja koje brojač treba da registruje pre nego što se generiše signal koji označava da su časovnik ili brojač završili rad.

  Zadata vrednost za časovnik može da se kreće u intervalu od 0 do +32767.

- **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 zadate vrednosti časovnik, odnosno brojač, završavaju svoj rad.

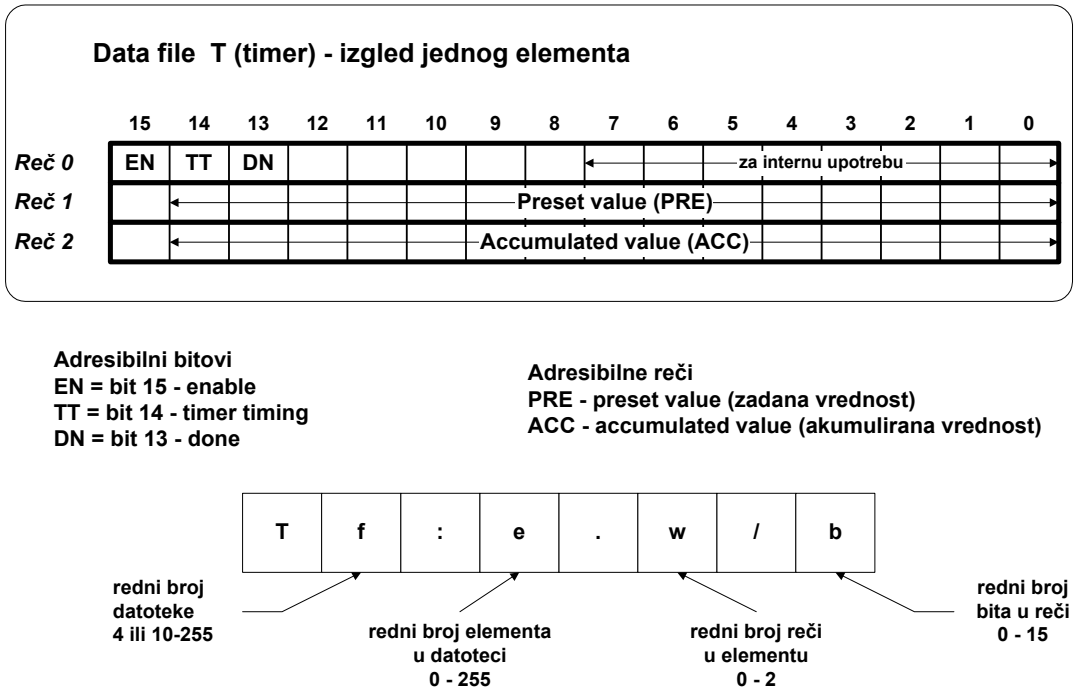
  Opseg dozvoljenih vrednosti za akumuliranu vrednosti isti je kao i za zatanu vrednost.

3.1.1 Datoteka podataka o časovniku (timer data file)

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 *zadatu vrednost* (PRE)
- **Reč 2** sadrži akumuliranu vrednost (ACC)



Sl. 3-3 Elemenat datoteke časovnika i adresiranje časovnika.

Na Sl. 3-3 prikazan je jedan elemenat datoteke časovnika, kojim se definiše jedan časovnik. Pored toga, prikazan je format adrese časovnika. Treba zapazi da *broj elementa* zapravo definiše jedan određeni časovnik unutar jedne datoteke časovnika. Svaki od tri bita stanja, kao i zadata i akumulirana vrednost mogu se posebno adresirati i to bilo na standardan način na koji se formira adresa u bilo kojoj datoteci podataka, bilo preko odgovarajućih simbola. To zapravo znači da su sledeće adrese međusobno ekvivalentne:

Tf:e.1 ili Tf:e.PRE
Tf:e.2 ili Tf:e.ACC
Tf:e.0/15 ili Tf:e/15 ili Tf:e/EN
Tf:e.0/14 ili Tf:e/14 ili Tf:e/TT
Tf:e.0/13 ili Tf:e/13 ili Tf:e/DN

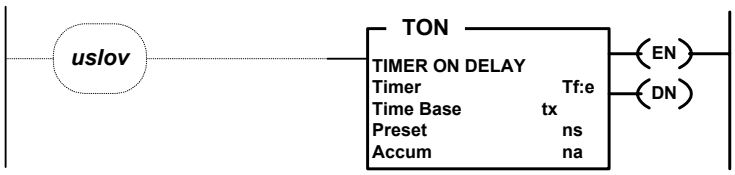
3.1.2 Naredbe časovnika

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

Potrebno je istaći da se sam časovnik 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 časovnika 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)***

TON naredba, grafički simbol i položaj urangu



Kao što je već rečeno, stavljanjem ove naredbe u leder program automatski se definiše prva vrsta časovnika 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 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* (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, 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.

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

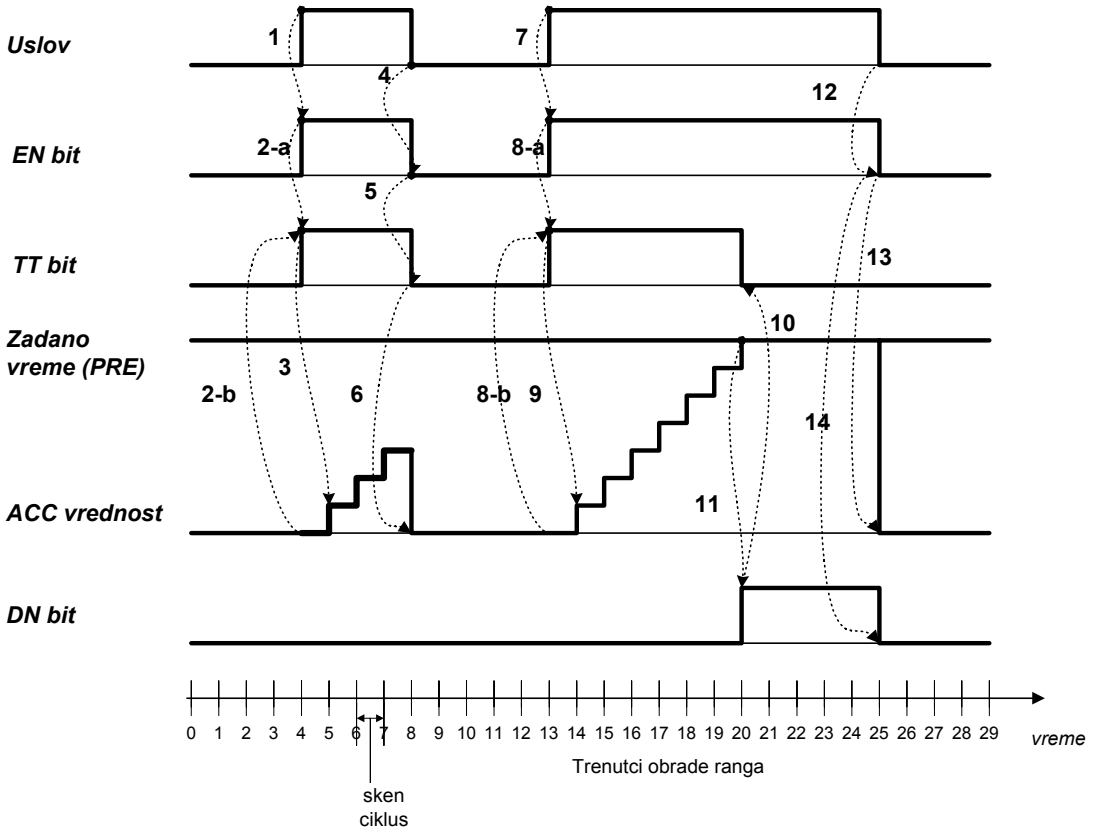
DN - Timer done bit se postavlja na 1 kada je ACC ≥ PRE. On se resetuje na 0 kad *uslov* u rangu postane *neistinit*.

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 časovnika potrebno zapaziti nekoliko činjenica. Pre svega **časovnik 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 časovnika *omogućen* (*enable*). Kada je *uslov neistinit*, EN bit ima vrednost 0, što znači da je rad časovnika *onemogućen*. Međutim, činjenica da EN bit ima vrednost 1 ne mora da znači da časovnik 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 časovnika** indicira TT bit. Naime, taj bit je postavljen na 1 za svo vreme za koje *časovnik aktivno meri vreme* (*timer timing*), i postavlja se na 0 kada časovnik ne radi. Konačno, kada je vrednost DN bita 1, onda to znači da je časovnik *završio* (*done*) svoj posao, tj. izmerio zadato vreme. Pri tome, DN bit ne govori o tome **kada** je časovnik završio sa poslom, jer će on ostati na vrednosti 1 sve dok *uslov* ne postane *neistinit.* Vremenski dijagram rada časovnika ilustrovan je na Sl. 3-4.

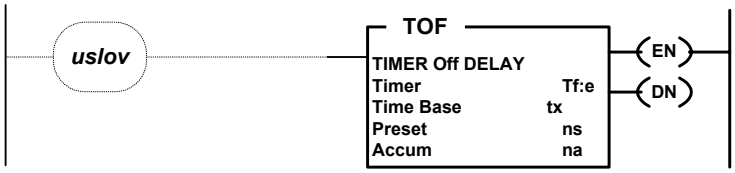
Stanje časovnika se može resetovati posebnom *RES naredbom*, o čemu će kasnije biti više reči.



Sl. 3-4 Vremenski dijagram izvršavanja TON naredbe.

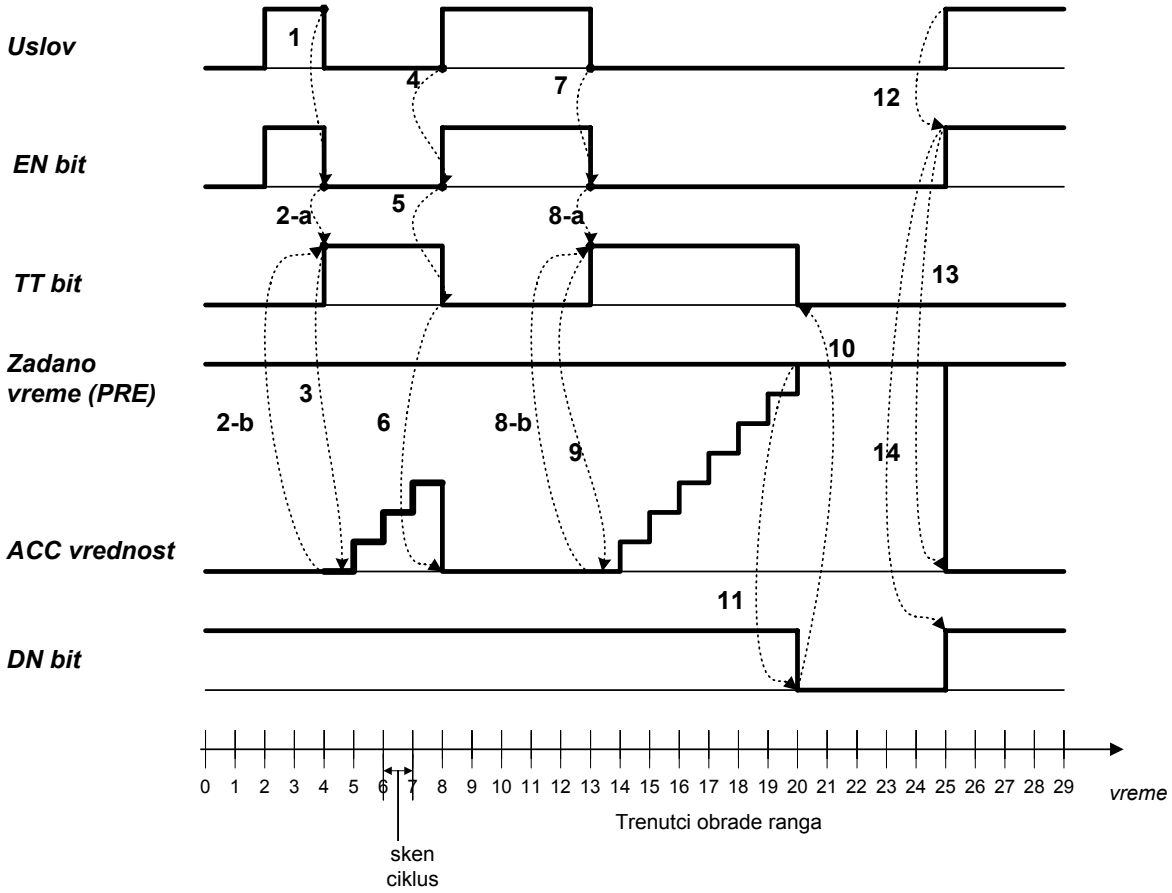
- ***Timer off-delay (TOF)***

TOF naredba, grafički simbol i položaj urangu



Ovom naredbom se definiše druga vrsta časovnika 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.

TOF naredba započinje rad časovnika za vreme onog programskog sken ciklusa u kome *uslov* u rangu u kome se naredba nalazi prvi put postaje ne*istinit* (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, časovnik prekida svoj rad. 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 izmerio zahtevano vreme ili ne.



Sl. 3-5 - Vremenski dijagram izvršavanja TOF naredbe

Bitovi stanja časovnika 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*.

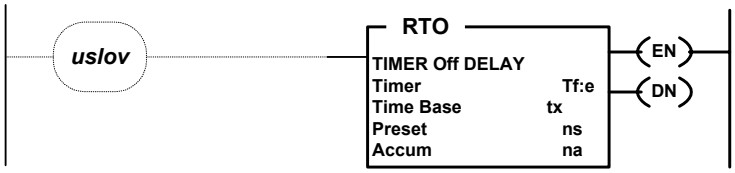
TT - Timer timing bit se postavlja na 1 kada je *uslov neistinit* i pri tome je ACC ≤ PRE. One se resetuje na nulu kada *uslov* postane *istinit* ili kada se DN bit resetuje.

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

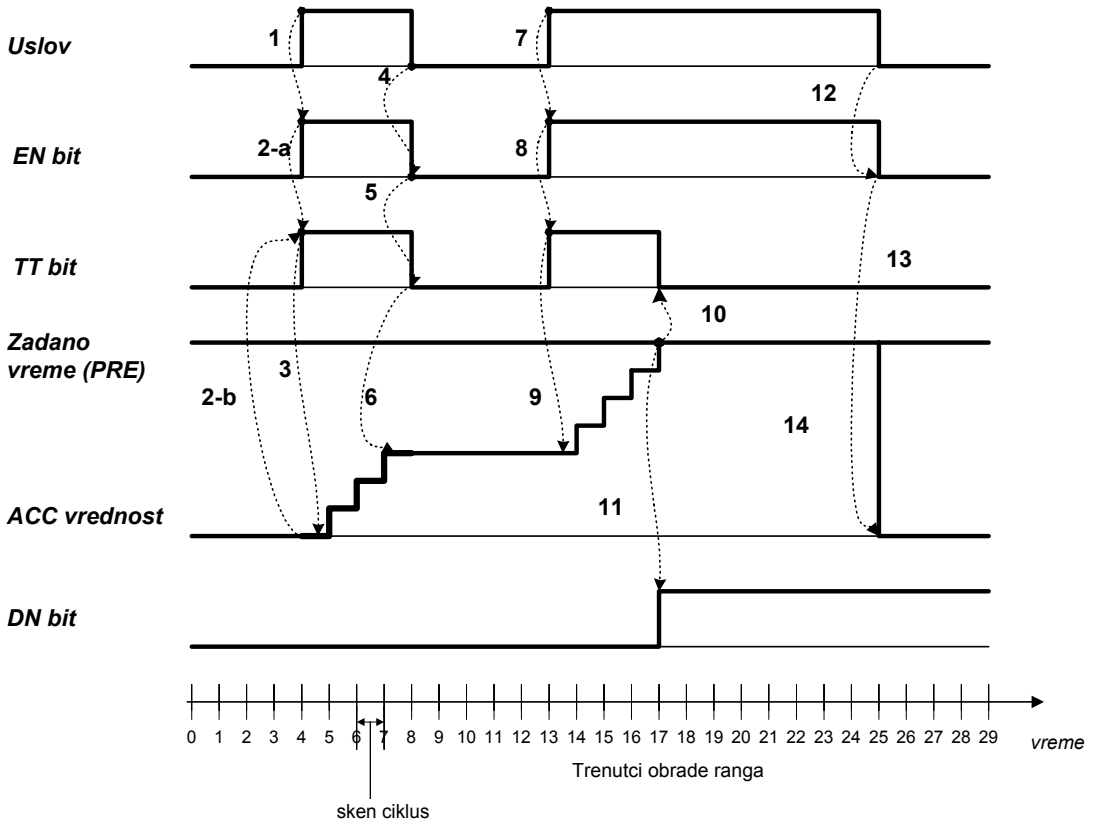
Pri tome, DN bit ne govori o tome **kada** je časovnik završio sa poslom, jer će on ostati na vrednosti 0 sve dok *uslov* ne postane *istinit.* Vremenski dijagram rada časovnika ilustrovan je na Sl. 3-5.

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

RTO naredba, grafički simbol i položaj urangu



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



Sl. 3-6 - Vremenski dijagram izvršavanja RTO naredbe

RTO naredba razlikuje se od TON naredbe samo po tome što se akumulirana vrednost ne resetuje, već zadržava i onda kada *uslov* postane *neistint*. Drugim rečima, ovaj časovnik počinje da radi kada *uslov* postane *istinit*, i nastavlja sa radom povećavajući akumuliranu vrednost sve dok je *uslov istinit*. Kada *uslov* postane *neistinit*, časovnik **prekida rad**, ali se akumulirana vrednost pri tome ne menja. To znači da će kada *uslov* ponovo postane *istinit*,

časovnik nastaviti sa radom i prethodno izmerenom vremenu (ACC) dodavati nove vrednosti. Na taj način ovaj časovnik omogućuje da se kumulativno mere intervali vremena u kojima je *uslov* bio *istinit* (Sl. 3-6).

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

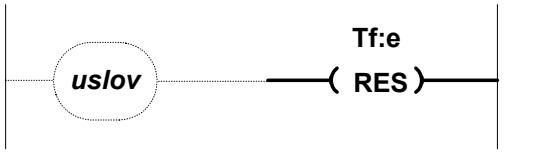
DN - Timer done bit se postavlja na 1 kada je ACC ≥ PRE (časovnik je izmerio zadato vreme). On se resetuje na 0 pomoću posebne RES naredbe.

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, grafički simbol i položaj u rangu



RES 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 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.

3.1.3 Način rada časovnika

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.

Ukoliko časovnik radi sa vremenskom bazom od 1 sekunde obrada časovnika je donekle složenija. Ovde se, naime i dalje koristi interni časovnik koji se ažurira na svakih 0,01 sekundi, ali se pri tome u toku obrade ranga akumulirana vrednost ažurira samo ako je akumulirana vrednost veća ili jednaka od 1 sekunde. Pri tome se akumulurina vrednost uvećava za 1, dok se eventualni ostatak vremena pamti u internom brojaču i na njega se

dodaju sledeći inkrimenti od po 0,01 sekunde. Postupak ažuriranja akumulirane vrednosti je takav da se može očekivati da će časovnik raditi ispravno ako sken ciklus ne traje duže od 1,5 sekundi. Naravno, i ovde se problem cikulusa dužeg trajanja može prevazići stavljanjem naredbe časovnika na više mesta u programu.

Potrebno je da se naglasi da je pri korišćenju časovnika neophodno da se posebna pažnja posveti naredbama za skok. Naime, i ako je trajanje sken ciklusa u dozvoljenim granicama, može se desiti da se nekom od naredbi za skok u jednom ili više suskcesivnih sken ciklusa preskoči rang koji sadrži časovnik. Jasno je da se u tom slučaju neće vršiti ažuriranje akumulirane vrednosti. To nadalje znači da je neophodno da se obezbedi da u slučaju bilo kakvog programskog skoka, naredba za časovnik ne bude isključena iz obrade u periodu koji je duži od maksimalno dozvoljenog vremena.

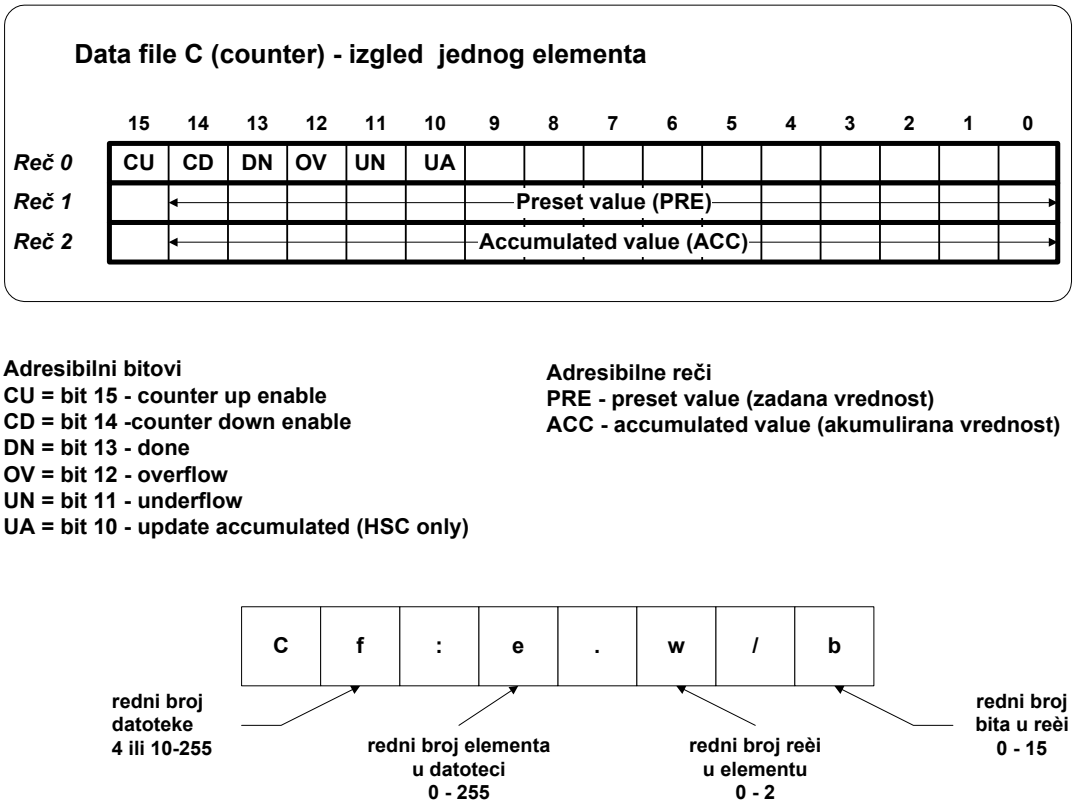
Tačnost časovnika je pojam koji se odnosi na dužinu vremenskog intervala koji protekne od trenutka kada se časovnik uključi do trenutka kada DN bit indicira da je merenje vremena završeno. Za časovnike koji rade sa vremenskom bazom od 0,01 sekunde tačnost je u granicama od ±0,01s sve dok sken ciklus ne traje duže od 2,5 sekunde. Časovnici koji rade sa vremenskom bazom od 1 sekunde zadržavaju svoju tačnost ukoliko je programski sken kraći od 1.5 sec.

Neophodno je da se istakne, međutim, da tačnost rada časovnika ne implicira da će i neki događaj koji je vezan sa časovnikom da bude aktiviran sa istom tačnošću. Aktiviranje događaja se ostvaruje ispitivanje DN bita. U najvećem broju slučajeva ovaj uslov se ispituje jedanput u okviru sken ciklusa. To nadalje znači da je tačnost aktiviranja događaja određena trajanjem jednog sken ciklusa.

3.2 Realizacija brojača

3.2.1 Datoteka podataka o brojaču (counter data file)

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.



Sl. 3-7 Elemenat datoteke brojača i adresiranje brojača.

Svakom brojaču pridružuju se po jedan *element* u odgovarajućoj datoteci (Sl. 3-7). *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 *zadatu vrednost* (PRE)
- **Reč 2** sadrži akumuliranu vrednost (ACC)

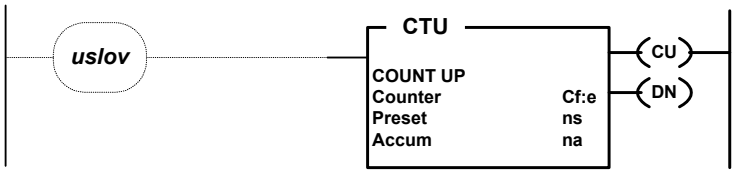
3.2.2 Naredbe brojača

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 –32767, i postavlja *underflow bit* (UN) kad pređe –32767.

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

CTU naredba, grafički simbol i položaj urangu



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

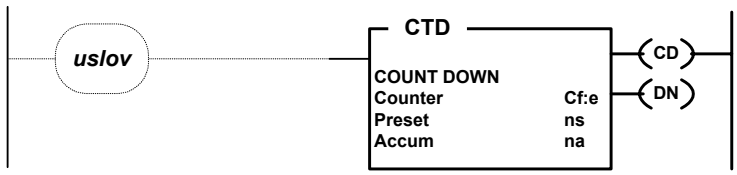
OV - Count up overflow bit se postavlja na 1 kada akumulirana vrednost (ACC) prelazi sa 32767 na –32768 (u binarnoj aritmetici drugog komplementa sa 16-bitnom reči važi: 32767+1= –32768), i nastavlja brojanje unapredi.

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)***

CTD naredba, grafički simbol i položaj urangu



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.

3.3 Primeri korišćenja časovnika i brojača

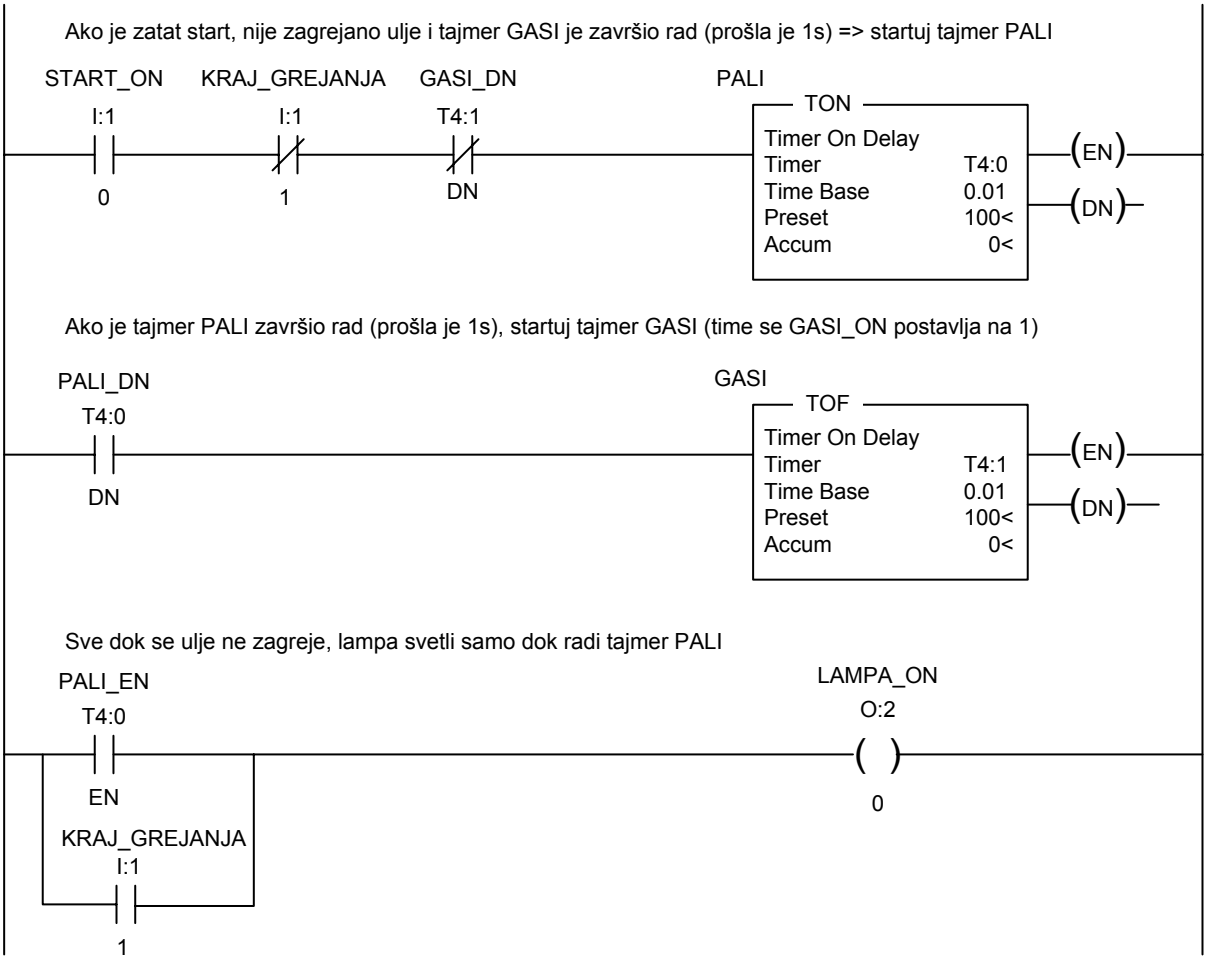
Primer: Paljenje i gašenje signalne lampe

Zadatak: Predpostavlja se da postoji neki hidraulični mehanizam za čiji rad je neophodno da se ulje zagreje do određene temperature. Pri tome se postizanje zadate temperature detektuje pomoću jednog termo-prekidača koji daje binarni signal kada temperatura postugne željenu

vrednost. Za vreme procesa zagrevanja ulja neophodno je da se signalna lampa pali i gasi (blinka), čime se indicira da se ulje zagreva. Kada ulje dostigne željenu temperaturu, signalna lampa počinje stalno da svetli. Sistem se pušta u rad pritiskom na jedan dvopoložajni prekidač.

Rešenje: Predpostavimo da kontroler ima jedan osmobitni diskretni ulazni modul smešten u slotu 1, i jedan osmobitni izlazni modul smešten u slotu 2. Neka je nadalje prekidač za puštanje sistema u rad vezan za nulti pin ulaznog modula, tako da ima adresu I:1/0 i neka je njegovo simboličko ime START_ON. Indikator temeperature je vezan za prvi pin, tako da je adresa signala I:1/1, i pridruženo simboličko ime KRAJ_GREJANJA. Signalna sijalica je vezana za pin 0 izlaznog modula (adresa O:2/0) i dodeljeno joj je simbolično ime LAMPA_ON.

Jedno moguće rešenje postavljenog zadatka prikazano je na Sl. 3-8.

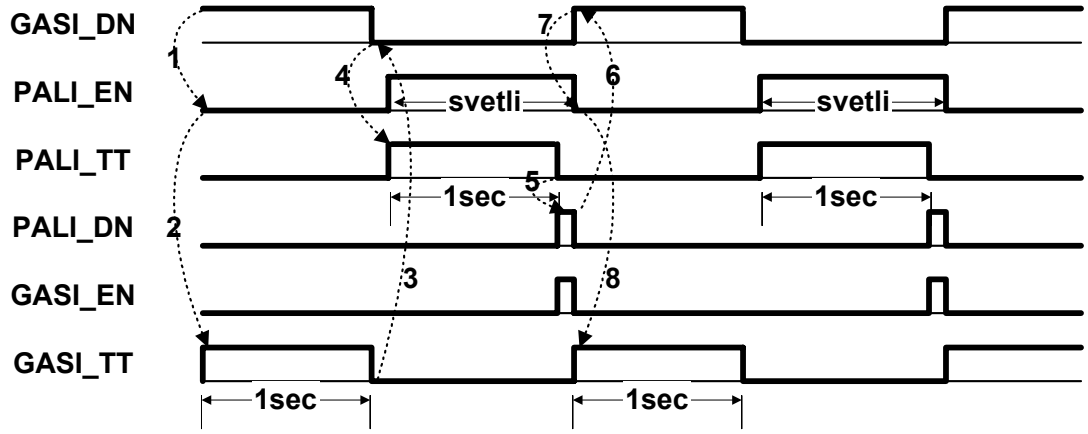


Sl. 3-8 Leder program za paljenje i gašenje sijalice.

Paljenje i gašenje signalne lampe može se ostvariti korišćeme dva časovnika od kojih je jedan TON tipa (simboličko ime PALI), a drugi TOF tipa (simboličko ime GASI) i koji su vezani tako da, za sve vreme za koje traje grejanje ulja, jedan od njih radi, a drugi ne radi. Čim jedan odbroji zadato vreme (ovde je usvojeno da to bude 1 sec), on se zaustavlja i aktivira se drugi. Ovo se postiže tako što se svaki od časovnika aktivira pomoću DN bita drugog časovnika. To je ujedno i razlog, zašto časovnici ne mogu biti istog tipa (oba TON ili oba TOF). Naime, u tom slučaju ne bi bilo moguće da se otpočne sa radom, jer bi na početku

oba DN bita imala istu vrednost (0 ili 1). Vremenski dijagram promene odgovarajućih bitova dat je na Sl. 3-9.

Sve dok se ne dostigne željena temperatura, bit označen kao KRAJ_GREJANJA ima vrednost 0, što znači da sijalica svetli samo za ono vreme za koje radi časovnik PALI, odnosno za koje je njegov EN bit postavljen na 1.



Sl. 3-9 Vremenski dijagram promene indikatorskih bitova časovnika.

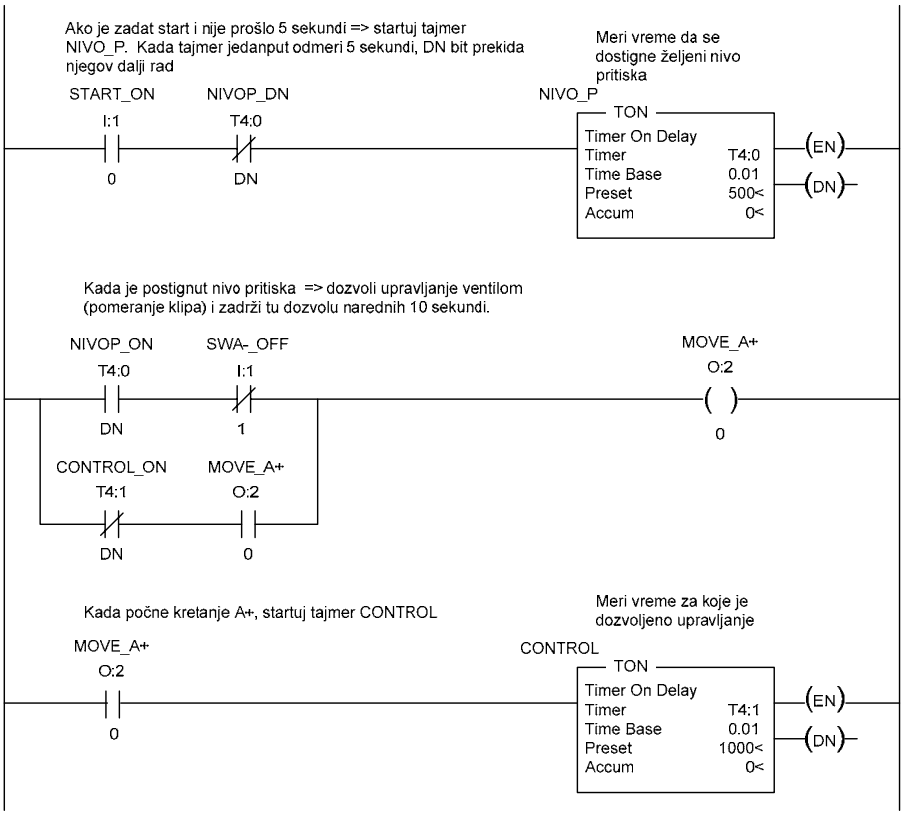
Primer 2: Regulacija protoka vazduha

Zadatak: Posmatra se neki ventil čijim radom se upravlja pomoću jednosmernog solenoida (A). Pri tome, kada se aktivira sistem, pritisak vazduha dostiže željeni nivo u roku od 5 sekundi. Nakon toga, ventil se otvara (pomeranjem klipa solenoida) i ostaje otvoren u vremenu od 10 sekundi. Posle isteka 10 sekundi, opruga solenoida vraća klip u početni položaj i zatvara ventil.

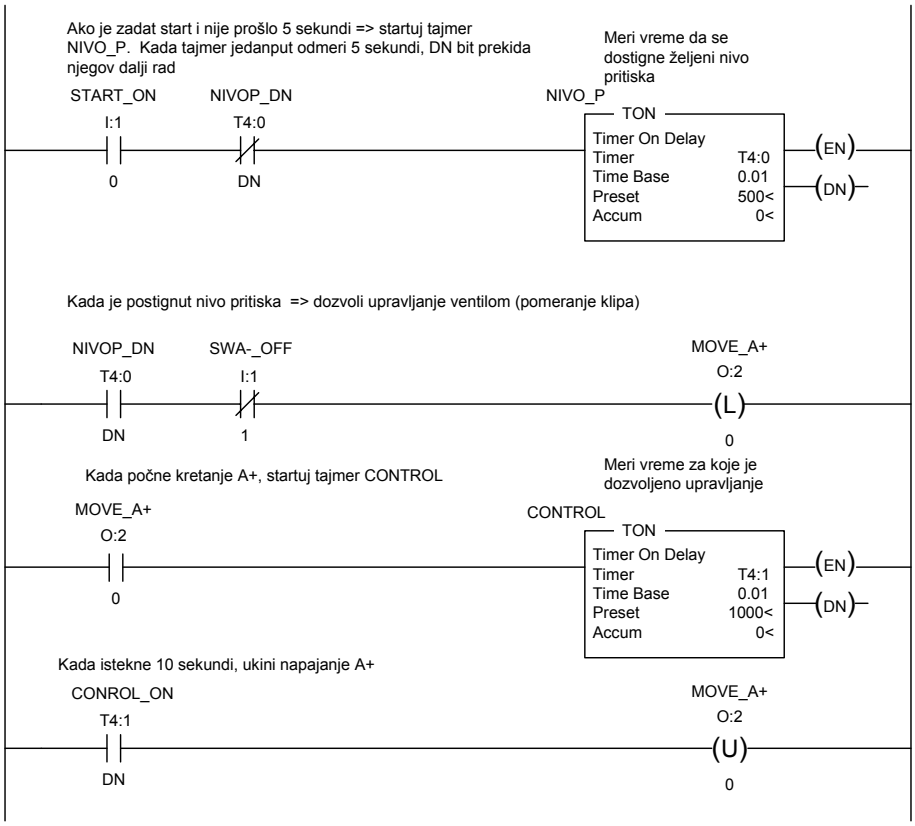
Rešenje: Dva moguća rešenja postavljenog problema, od kojih jedno koristi izlaznu naredbu sa lečovanjem, a drugo postupak samodržanja, prikazana su na Sl. 3-10 i Sl. 3-11. Predpostavljeni način vezivanja signala vidi se iz samih programa. Pri tome je usvojeno da je prekidač koji ukazuje da je klip u krajnjem uvučenom položaju realizovan kao *normalno zatvoren*. Imajući u vidu da ovaj prekidač daje binarni signal koji ima vrednost 1 onda kada nije pritisnut, odnosno kad klip nije potpuno uvučen, a poštujući princip da simboličko ime promenljive treba da ukazuje na stanje u kome se ona nalazi kada ima vrednost 1, ovom prekidaču je pridruženo simboličko ime SWA-_OFF.

Potrebno je obratiti pažnju na činjenicu da program u kome se koristi samodržanje ima jedan rang manje i to zato što se kod njega u okviru istog ranga ostvaruje i aktiviranje i deaktiviranje signala A+. Naime, aktivacija se postiže tako što se čeka da se postigne nivo pritiska (odnosno da istekne 5 sekundi – DN bit časovnika NIVO_P ima vrednost 1) i tada se, ako je klip uvučen (*nije* SWA-_OFF), daje komanda kojom se signal vodi na priključak A+ (simboličko ime MOVE_A+). Nadalje se ovaj signal zadržava pomoću ispitivanja stanja MOVE_A+ (samodržanje) sve dok ne istekne 10 sekundi, što se proverava pomoću DN bita časovnika CONTROL.

U programu u kome se koristi lečovanje, aktivacija solenoida se postiže u jednom rangu i to na isti način na koji se to ostvaruje kod samodržanja, dok se deaktivacija, prirodno, mora naći u zasebnom rangu i uslovljena je stanjem DN bita časovnika CONTROL.



Sl. 3-10 Regulacija protoka vazduha sa samodržanjem.



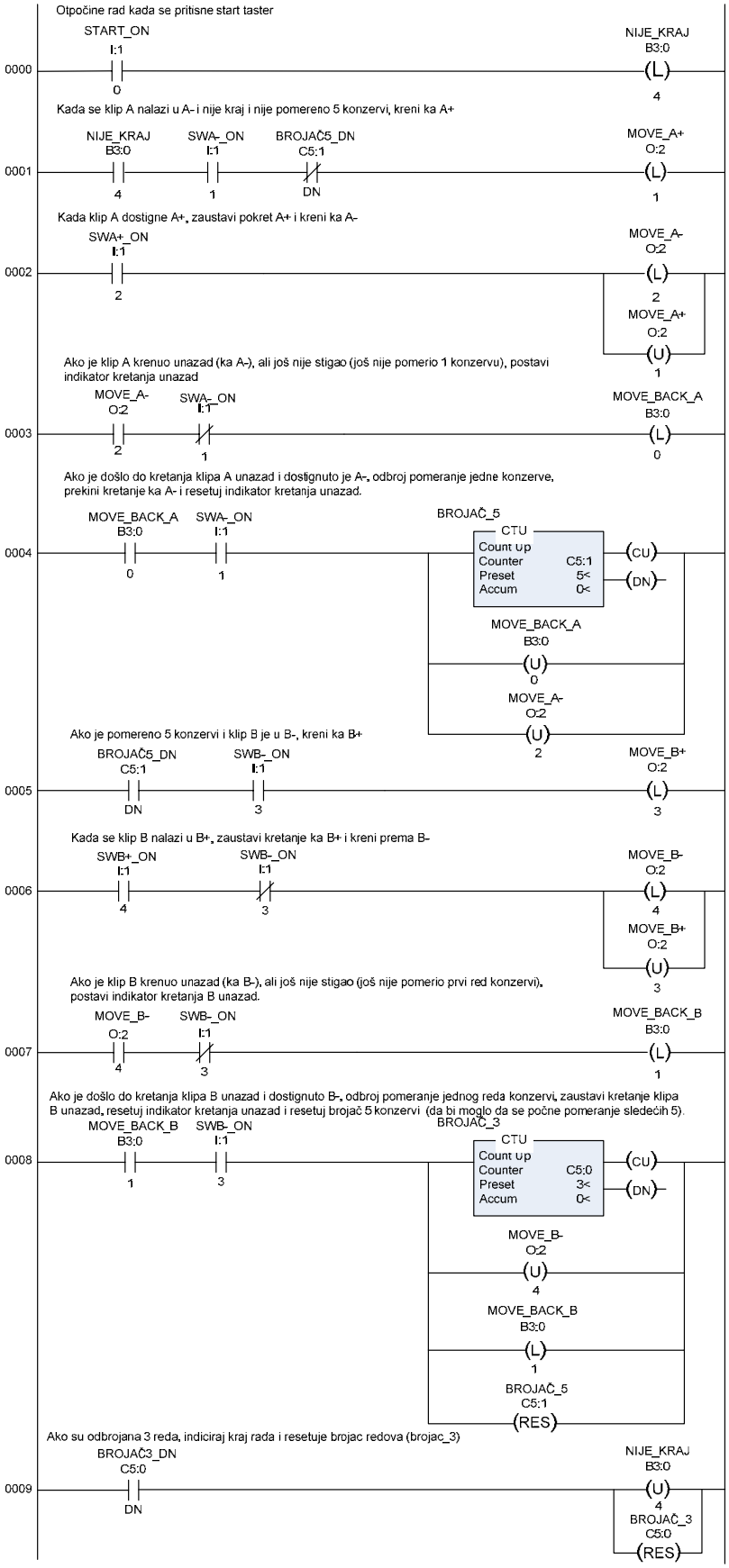
Sl. 3-11 Regulacija protoka vazduha sa lečovanjem.

Primer 4: Mašina za pakovanje

Zadatak: Posmatra se jedna mašina koja pakuje konzerve u kutiju i to tako da u kutiji ima tri reda konzervi, pri čemu u svakom redu ima po 5 konzervi. Mašina radi tako što se pomoću jednog klipa (A) konzerve koje dolaze preko pokretne trake guraju, jedna po jedna, na pomoćno postolje. Kada se na postolje smesti 5 konzervi, onda se one, sve zajedno, pomoću drugog klipa (B) gurnu u kutiju. U skladu sa time, za pakovanje jedne kutije potrebno je da se ostvari sledeća sekvenca pomeranja klipova:

[(A+ A-)x5 B+ B-]x3

Rešenje: Moguće rešenje dato je na Sl. 3-12. Pri tome je predpostavljeno da su oba klipa dvosmerna, i da su svi granični prekidači *normalno otvoreni*. Koriste se dva brojača: BROJAC_5 koji broji konzerve u jednom redu, i BROJAC_3 koji broji redove. Pomoćni bit NIJE_KRAJ, postavlja se na 1 pritiskom na taster START, a vraća na 0 kad BROJAC_3 odbroji do kraja (do 3). Za vreme dok je NIJE_KRAJ=1, obavalja se sekvenca pomeranja klipova. Klip A se izvlači/uvlači rangovima 1 i 2. Rang 3 formira ´impuls´ za brojanje pomerenih konzervi. U rangu 4 broje se pomerene konzerve. Rangovi 5 i 6 izvlače/uvlače klip B nakon svakog popunjenog reda (onda kad je BROJAC_5 izbrojao do 5). Rang 7 formira ´impuls´ za brojanje redova. Redovi se broje u rangu 8. Takođe, u ovom rangu, resetuje se brojač konzervi (BROJAČ_5). U rangu 9, ispituje se da li su formirana 3 reda, i ako jesu, bit NIJE_KRAJ postavlja se na 0 i resetuje BROJAC_3, čime su pripremljeni uslovi, da se nakon novog pritiska na taster START, ponovi celokupna sekvenca.



Sl. 3-12 Mašina za pakovanje (realizacija korišćenjem lečovanja).

4 Naredbe za operacije nad podacima

U realizaciji različitih algoritama često je potrebno da se izvrše određena izračunavanja, da se prenesu odgovarajuće poruke ili da se u zavisnosti od vrednosti nekih parametara promeni algoritam obrade. U osnovi svih navedenih aktivnosti nalaze se *promenljive – podaci* koji predstavljaju *operande* ili rezultate u različitim matematičkim ili logičkim *operacijama*.

- ***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 *celobrojne* vrednosti (*integers*) ili *decimalni brojevi prikazani u formatu pokretnog zareza* (*floating point*). Različiti tipovi numeričkih podataka smeštaju se u *datoteke podataka* odgovarajućeg tipa.

  U principu, operandi mogu biti promenljive iz bilo koje datoteke. Potrebno je uočiti, međutim, da iako se dozvoljava korišćenje bit-adresibilnih datoteka (B,I,O), podaci smešteni u njima se u ovim operacijama mogu koristiti samo kao cele reči (elementi), što znači da se operacija ne može izvoditi nad pojedinim bitovima. Pored toga, u datotekama časovnika i brojača (T i C) mogu se kao operandi koristiti samo druga i treća reč elementa koje predstavljaju akumuliranu vrednost (ACC) i zadanu vrednost (PRE). Konačno, kao operandi se mogu javiti i neke promenljive iz kontrolne datoteke (R). O značenju i ulozi ovih promenljivih biće reči kasnije.

  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 koristiti kao rezultat.

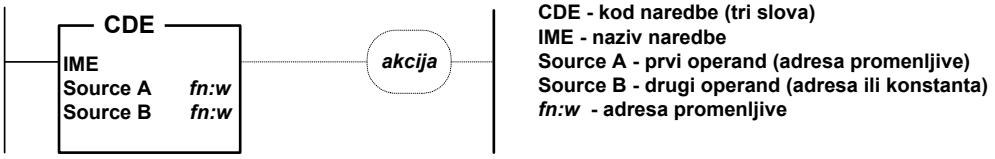
- ***operacije***

  Operacija koja treba da se izvrši nad operandima definiše se u okviru naredbe. Najveći broj ovih naredbi pojavljaju se kao *naredbe akcije*. Ovo je sasvim prirodno ako se ima u vidu da je glavna svrha ovih naredbi da se obavi neka aritmetička ili logička operacija nad operandima i dobijeni rezultat upamti kao odgovarajuća promenljiva. 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 njihovog izvođenja istinosna vrednost naredbe.

4.1 Naredbe za poređenje

Naredbe za poređenje su naredbe *uslova*. U okviru ovih naredbi proverava se istinosna vrednost relacije između dva operanda. Kao rezultat provere naredba dobija vrednost *istinit* ili *neistinit*. Jedna grupa naredbi za poređenje ima oblik kao što je to prikazano na Sl. 4-1. U tabeli T. 1 dat je pregled svih naredbi za poređenje iz ove grupe. P*rvi operand* je uvek *promenljiva*, dok *drugi operand* može biti ili *promenljiva* ili *programska konstanta*.

Naredbe za poređenje, grafički simbol i položaj u rangu



Sl. 4-1 Opšti izgled naredbe za poređenje.

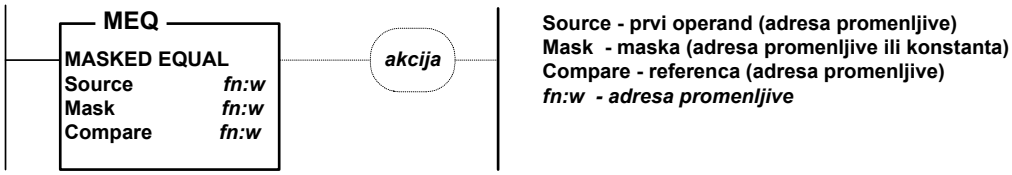
T. 1 Grupa naredbi za poređenje

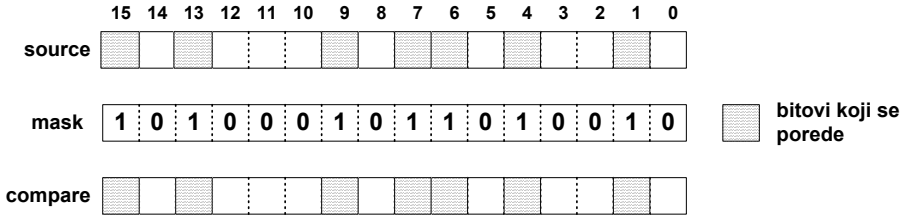
Kod naredbe	Ime naredbe	relacija	
		istinita ako je	neistinita ako je
EQU	Equal (jednako)	A = B	A ≠ B
NEQ	Not equal (nejednako)	A ≠ B	A = B
LES	Less than (manje)	A < B	A > B
LEQ	Less than or equal (manje ili jednako)	A ≤ B	A ≥ B
GRT	Greater than (veće)	A > B	A < B
GEQ	Greater than or equal (veće ili jednako)	A ≥ B	A ≤ B

Pored navedenih naredbe među naredbama za poređenje postoje i sledeće dve naredbe.

- **MEQ - masked comparison for equal (maskirano ispitivanje jednakosti*)***

MEQ naredba, grafički simbol, položaj u rangu i realizacija

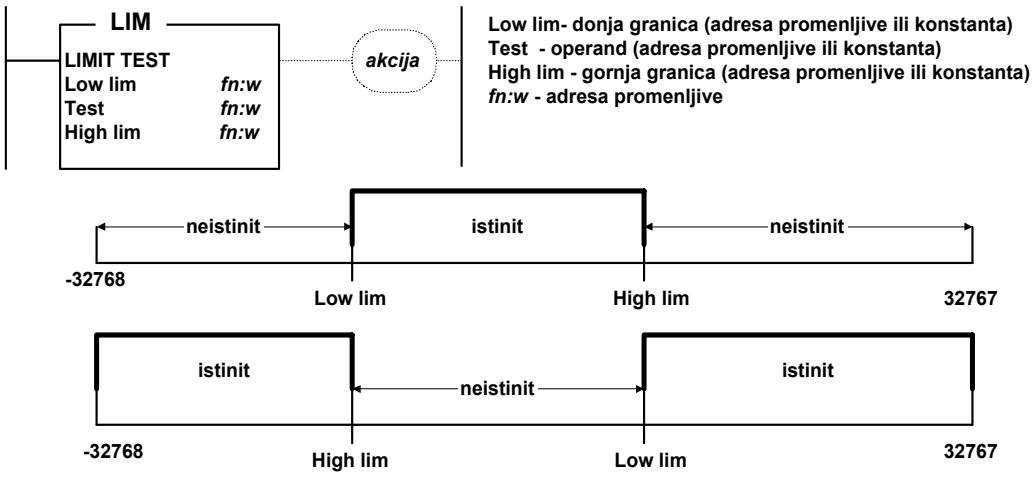




Ova naredba služi za poređenje delova pojedinih reči. Naime na položaju onih bitova koji ne učestvuju u poređenju (*maskirani bitovi*) u maski se stavljaju *nule*. Ostali bitovi maske, koji odgovaraju bitovima koji se porede (*nemaskirani bitovi*), se postavljaju na *1*. Ukoliko su bitovi *operanda* i *reference* koji nisu *maskirani* međusobno jednaki naredba ima vrednost *istinit*. U protivnom ona ima vrednost *neistinit*. Pri definisanju maske, pogodno je koristiti heksadecimalnu konstantu ili promenljivu.

- **LIM – Limit test (ispitivianje granica)**

LIM naredba, grafički simbol, položaj u rangu i realizacija



LIM 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. Potrebno je obratiti pažnju na činjenicu da "*donja granica*" može biti i veća od "*gornje granice*". U tom slučaju naredba je *istinita* ako se operand nalazi izvan granica ili na njima, a *neistinita* ako operand pripada intervalu koji određuju granice.

Ako je operand *test* konstanta, onda obe granice moraju biti adrese promenljivih. Međutim, ukoliko je *test* adresa promenljive, onda granice mogu biti bilo adrese promenljivih bilo konstante.

4.2 Matematičke naredbe

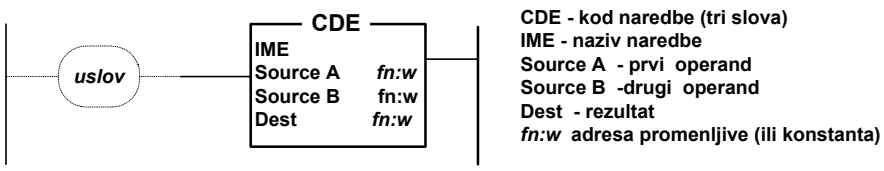
Kako im i samo ime kaže, *matematičke naredbe* služe za realizaciju različitih operacija nad operandima. Ove naredbe su naredbe *akcije* i u najvećem broju slučajeva imaju *dva operanda*. Izvršavanjem naredbe obavlja se zahtevana matematička operacija nad operandima i dobija rezultat čija se vrednost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

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

4.2.1 Aritmetičke i logičke binarne operacije

Opšti oblik naredbe za aritmetičke i logičke binarne operacije dat je na Sl. 4-2, dok je u tabeli T. 2 dat prikaz svih naredbi iz ove grupe.

Aritmetičke i logičke operacije, grafički simbol i položaj u rangu



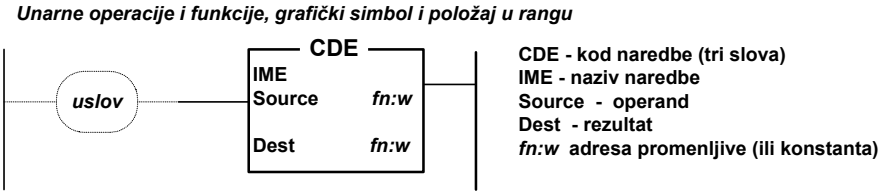
Sl. 4-2 Opšti oblik naredbe za aritmetičke i logičke binarne operacije.

T. 2 Pregled naredbi za aritmetičke i logičke binarne operacije.

Kod naredbe	Ime naredbe	Operacija	Postavlja indikatorske bitove			
			C – bit	V – bit (ako je S:2/14=0)	Z – bit	S – bit
ADD	Add (sabiranje)	d = a + b	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
SUB	Subtract (oduzimanje)	d = a – b	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
MUL	Multiply (množenje)	d = ab	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
DIV	Divide (deljenje)	d = a/b	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
XPY	X to the power of Y	$d = x^y$	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
AND	And (logičko "i")	$d = a \wedge b$ bit po bit	uvek 0	uvek 0	1 za d=0	1 za d<0
OR	Or (logičko "ili")	$d = a \vee b$ bit po bit	uvek 0	uvek 0	1 za d=0	1 za d<0
XOR	Exclusive OR (ekskluzivno "ili")	$d = a \underline{\vee} b$ bit po bit	uvek 0	uvek 0	1 za d=0	1 za d<0

4.2.2 Unarne operacije

U grupu unarnih operacija svrstane su i aritmetičke i logičke unarne operacije. Sve ove naredbe imaju isti opšti oblik (Sl. 4-3). Pregledsvih naredbi dat je u tabeli T. 3.



Sl. 4-3 Opšti oblik naredbe za unarne operacije.

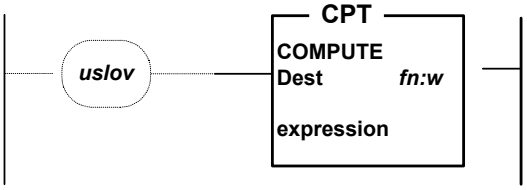
T. 3 Pregled naredbi za unarne operacije.

Kod naredbe	Ime naredbe	Operacija	Postavlja indikatorske bitove			
			C – bit	V – bit (ako je S:2/14=0)	Z – bit	S – bit
NEG	Negate (negacija)	d = -a	0 za d=0 ili V=1	1 za prekoračenje opsega (samo ako je a=-32768)	1 za d=0	1 za d<0
NOT	Not (komplement)	$d = \bar{a}$ bit po bit	uvek 0	uvek 0	1 za d=0	1 za d<0
DDV	Double divide (deljenje 32-bitnog celog broja iz mat. reg. sa 16-bitnim operandom)	d = reg/a (rezultat zaokružen)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
SQR	Square Root (kvadratni koren)	$\mathbf{d = \sqrt{a}}$	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
ABS	Absolute (apsolutna vrednost)	d =|a|	uvek 0	1 samo ako je a=-32768	1 za d=0	uvek 0
SIN	Sine	d = sin(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
COS	Cosine	d = cos(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
TAN	Tangent	d = tg(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
ASN	Arc Sine	d = arcsin(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
ACS	Arc Cosine	d = arccos(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
ATN	Arc Tangent	d = artg(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
LN	Natural log (prirodni logaritam)	d = ln(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0
LOG	Log to the base 10 (dekadni logaritam)	d = log(a)	uvek 0	1 za prekoračenje opsega	1 za d=0	1 za d<0

4.2.3 Složene matematičke naredbe

- **CPT – Compute (izračunavanje aritmetičkog izraza)**

CPT naredba, grafički simbol i položaj u rangu

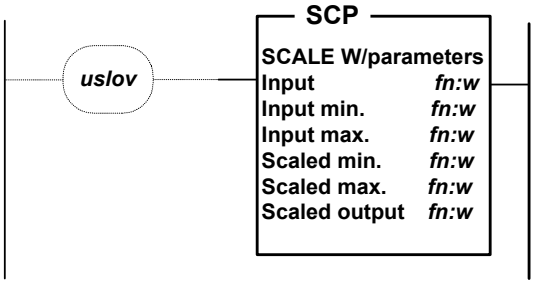


Dest - rezultat
expression - aritmetički izraz koji ima najviše 255 znakova raspoređenih u linijama. Svaka linija može imati najviče 28 znakova.
fn:w - adresa promenljive

Pri formiranju izraza koriste se promenljive, konstante i sledeći operatori: – ili NEG (unarni minus), +, –,* , | ili DIV, ** ili XPY, SQR, ABS, SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, NOT, AND, OR, XOR, TOD, FRD, DEG, RAD.

- **SCP – Scale with parameters (parametarsko skaliranje podatka)**

SCP naredba, grafički simbol i položaj u rangu



Input- ulazna promenljiva (operand)
Input min. - najmanja vrednost ulaza
Input max. - najveća vrednost ulaza
Scaled min. -najmanja vrednost rezultata
Scaled max. - najveća vrednost rezultata
Scaled output - rezultat
fn:w adresa celobrojne ili realne promenljive (može biti konstanta za min. i max. vrednosti)

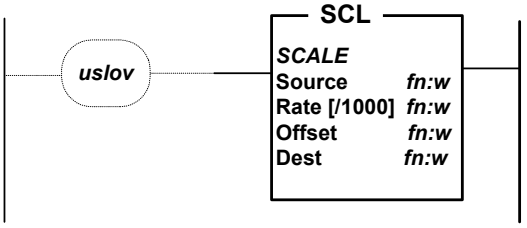
Ovom naredbom se ostvaruje linearno preslikavanje ulazne promenljive, prema relaciji

$$\mathbf{scaled\_output} = \frac{\mathbf{scaled\_max.} - \mathbf{scaled\_min.}}{\mathbf{input\_max.} - \mathbf{input\_min.}} \times (\mathbf{input} - \mathbf{input\_min.}) + \mathbf{scaled\_min.}$$

Potrebno je da se istakne da se u ovoj naredbi pojmovi "najmanja i najveća vrednost ulaza i rezultata" zapravo koriste samo za određivanje dve tačke kroz koje se provlači prava linija koja određuje preslikavanje. To znači da vrednost ulazne promenljive ne mora biti unutar intervala određenog sa (*input_min., input_max.*), niti da vrednost *input_min*. mora biti manja od *input_max*.

- **SCL - Scale data (skaliranje podatka)**

SCL naredba, grafički simbol i položaj u rangu



SCL - kod naredbe (tri slova)
SCALE - naziv naredbe
Source - operand
Rate - promenljiva ili konstanta u opsegu (-32768, 32767)
Offset - promenljiva ili konstanta
Dest - rezultat
***fn:w* adresa promenljive**

Ova naredba je slična naredbi SCP, jer se i pomoću nje ostvaruje linearno skaliranje promenljive. Razlika je samo u načinu definisanja parametara skaliranja. Skaliranje ulaza se izvodi prema sledećoj relaciji:

$$\mathbf{Dest} = \frac{\mathbf{Rate}}{\mathbf{1000}} \times \mathbf{Source} + \mathbf{Offset}$$

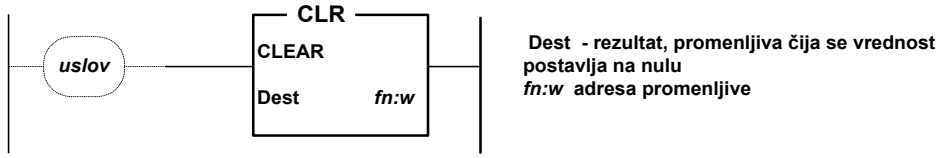
4.2.4 Naredbe za manipulaciju sa numeričkim podacima

Naredbe za manipulaciju sa podacima su skup naredbi kojima se definišu vrednosti promenljivih ili vrše 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.

4.2.4.1 Naredbe za postavljanje vrednosti

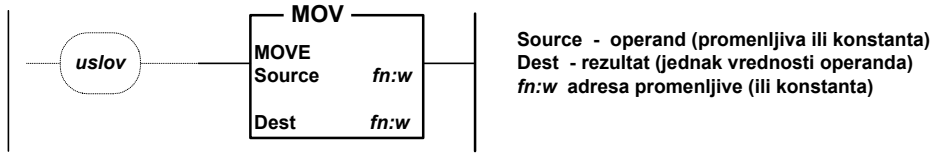
- **CLR – Clear (postavi na nulu)**

CLR naredba, grafički simbol i položaj u rangu



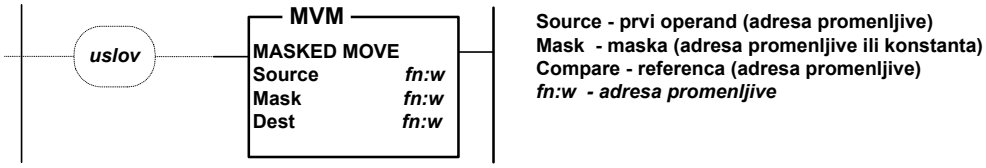
- **MOV – Move (postavljanje vrednosti promenljive)**

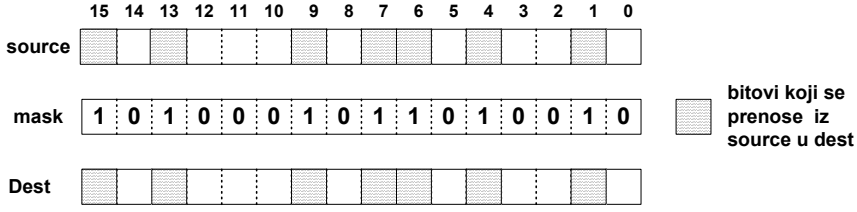
Mov naredba, grafički simbol i položaj u rangu



- **MVM – Masked move (postavljanje vrednosti pojedinih bitova)**

MVM naredba, grafički simbol, položaj u rangu i realizacija





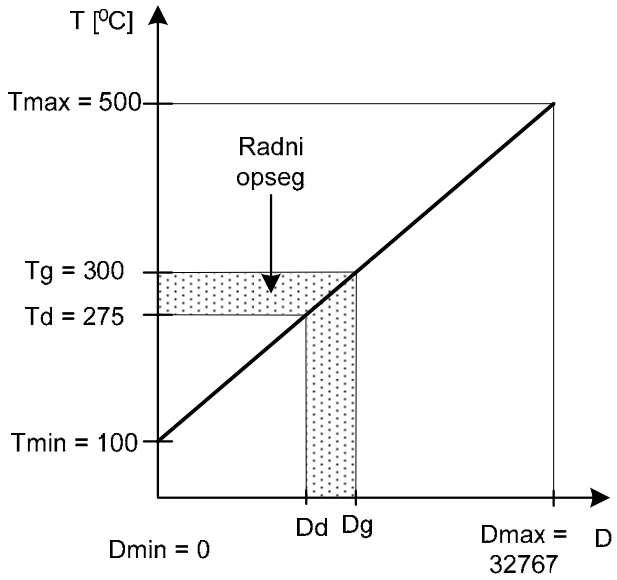
Maskiranim bitovima, koji se ne prenose u *dest* u maski odgovaraju vrednopsti 0, dok nemaskiranim bitovima odgovaraju vrednosti 1.

4.3 Primeri korišćenja naredbi za operacije nad podacima

Primer: Adresiranje, ispitivanje opsega i skaliranje analognog ulaza.

Zadatak: Analogni modul NI4 smešten je u slotu 1 modularnog PLC kontrolera. Temperatoruni senzor sa naponskim izlazom opsega 0-10V povezan je na drugi ulazni kanal analognog modula. Izlazni napon temeraturnog senzora direktno je srazmeran temperaturi iz opsega 100^0C (0V) – 500^0C (10V). Temperatura procesa mora ostati u opsegu 275^0C-300^0C. Ako je temperatura izvan ovog opsega, treba aktivirati alarm koji je vezan za pin 0 diskretnog izlaznog modula smeštenog u slotu 2. Vrednost temperature izraženu u fizičkim jedinicama (stepenima Celzijusa) treba smestiti u reč N7:0 radi dalje obrade.

Rešenje: Analogi modul NI4 obavalja 16 bitnu A/D konverziju za opseg ulaznog napona -10V - +10V. Rezultujuća izlazna digitalna reč dostupna je u formatu dvojičnog komplementa. Pošto je u konkretnom primeru ulazni napon unipolaran, opseg izlaznih reči biće: od 0 (za 0V) do $32767=2^{15}$-1 (za 10V). Između digitalnih reči i temperature postoji linearna zavisnost, kao što je prikazano na Sl. 4-4. Na dijagramu sa Sl. 4-4 takođe je naznačen radni temperaturni opseg. Najnižoj temperaturi radnog opsega ogovara digitalna reč Dd (donja granica), a najvišoj digitalna reč Dg (gornja granica).



Sl. 4-4 Zavisnost između ulazne digitalne reči i temperature.

Vrednosti Dd i Dg možemo izračunati na sledeći način:

$$T = \frac{T\max - T\min}{D\max - D\min}(D - D\min) + T\min \text{ - jednačina prave T(D)}$$

za konkretne vrednosti dobijamo:

$$T = \frac{500-100}{32767-0}(D-0)+100 = \frac{400}{32767}D + 100\,[^0C]$$

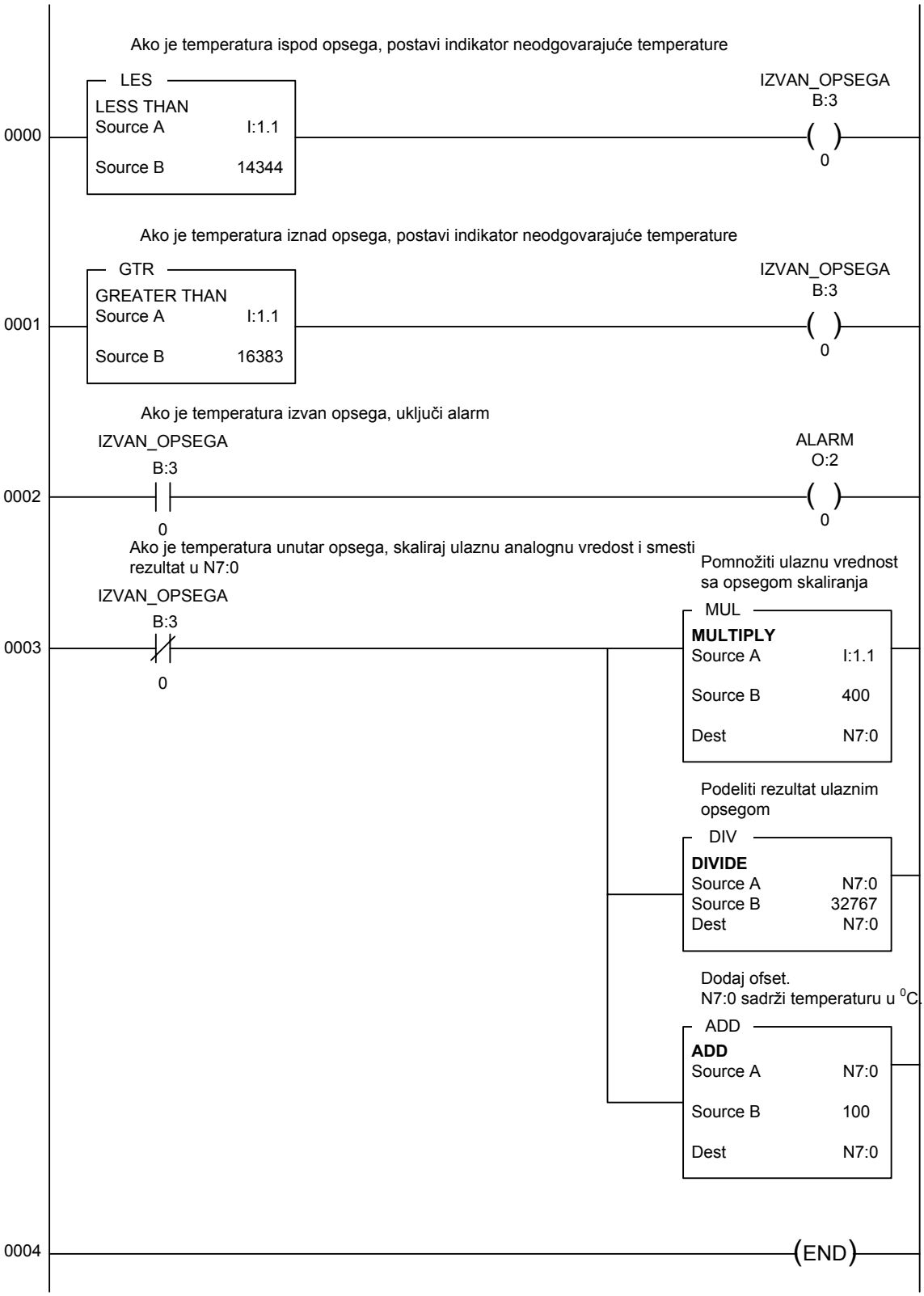
odnosno:

$$D = (T - 100^0C)\frac{32767}{400}\text{, odakle sledi}$$

$Dd = D(275) = 14344$

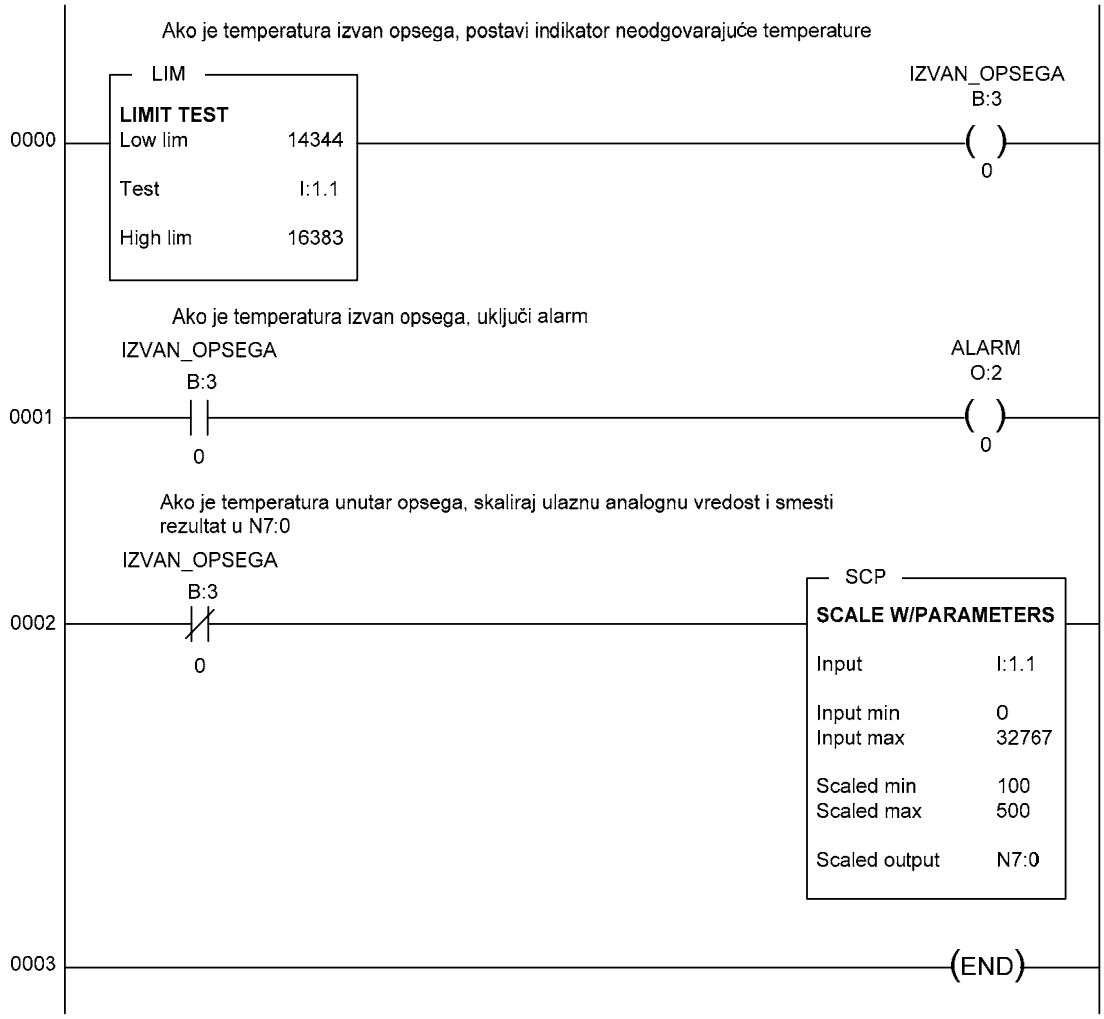
$Dg = D(300) = 16383$

Na Sl. 4-5 prikazan je lader program koji ispituje da li ulazna digitalna reč pripada radnom opsegu; aktivira alarm ako je temperatura izvan opsega, odnosno, obavlja konverziju ulazne digitalne reči u ^{0}C, ako je temperatura u radnom opsegu. Lader program sa Sl. 4-5 koristi naredbe poređenja i standardne matematičke naredbe.



Sl. 4-5 Skaliranje analogne ulazne vrednosti (relizacija pomoću standardnih aritmetičkih naredbi).

Leder program se može pojednostaviti ako se koristi naredba za ispitivanje granica (LIM) i naredba za linearno parametarsko skaliranje podataka (SCP), kao što je prikazano na Sl. 4-6.



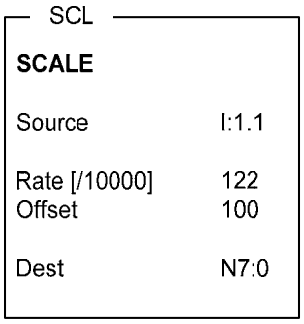
Sl. 4-6 Skaliranje analogne ulazne vrednosti (relizacija pomoću LIM i SCP naredbi).

Umesto naredbe SCP, za linearno skaliranje ulazne analogne vrednosti možemo koristiti i naredbu SCL. Prethodno je potrebno odrediti parametre **Rate** i **Offset** naredbe SCL:

Rate = (400/32767)x10000 = 122

Offset = 100

Na Sl. 4-7 je prikazan blok naredbe SCL sa uvršćenim konkretnim parametrima.



Sl. 4-7 SCL naredba koja odgovara SCP naredbi iz leder dijagrama sa Sl. 4-6

Primer: Skaliranje izlazne analogne veličine

Zadatak: Neka je u slotu 2 modularnog PLC kontrolera prisutan analogni modul NIO4I. Aktuator električnog ventila povezan je na izlazni kanal 0 analognog modula. Aktuator se pobuđuje strujom opsega 4mA – 20mA. Otvor ventila je proporcionalan struji, tako da struja od 4mA potpuno zatvara ventil (otvorenost ventila od 0%), a struja od 20mA potpuno otvara ventil (otvorenost ventila od 100%). Aktuator ventila ne sme biti pobuđivan strujom koja je izvan zadatog opsega. Nacrtati leder dijagram za određivanje izlazne analogne vrednosti, pod pretpostavkom da je procenat otvorenosti ventila (0-100%) dostupan u registru N7:0.

Rešenje: Analogni modul NIO4I poseduje dva analogna strujna izlaza. Opseg izlazne struje je -21mA - +21mA. Ulazne digitalne vrednosti su 16 bitne u formatu dvojičnog komplementa. U konkretnom primeru, izlazna struja je ograničena na pozitivan podopseg 4mA-20mA. Granicama dozvoljenog opsega izlazne struje odgovaraju sledeće digitalne vrednosti:

4mA -> (32767 / 21)x4 = 6242 (*scaled_min*)
20mA -> (32767 / 21)x20 = 31208 (*scaled_max*)

Zavisnost skalirane izlazne vrednosti, *scaled_value* (digitalna vrednost koja se šalje analognom modulu) od ulazne vrednosti, *input_value*, (procenat otvorenosti ventila) prikazana je na Sl. 4-8, a određena je sledećom jednačinom:

scaled_value = (*input_value* x *slope*) + *offset*,

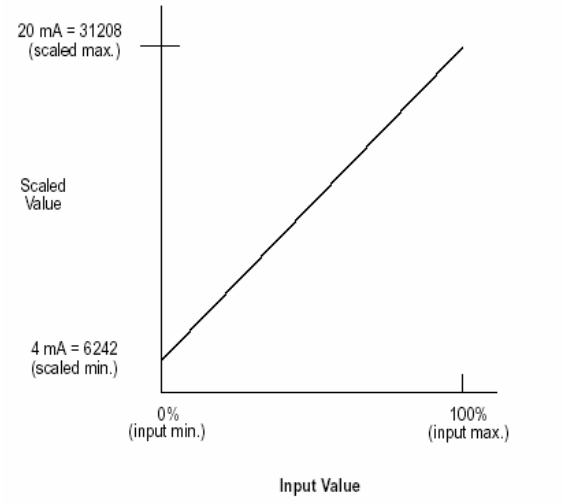
gde je *slope* nagib prave određene krajnjim tačkama ulaznog i izlaznog opsega:

slope = (*scaled_max* – *scaled_min*)/(*input_max* – *input_min*)
= (31208 – 6242)/(100-0) = 24966/100

dok je *offset*:

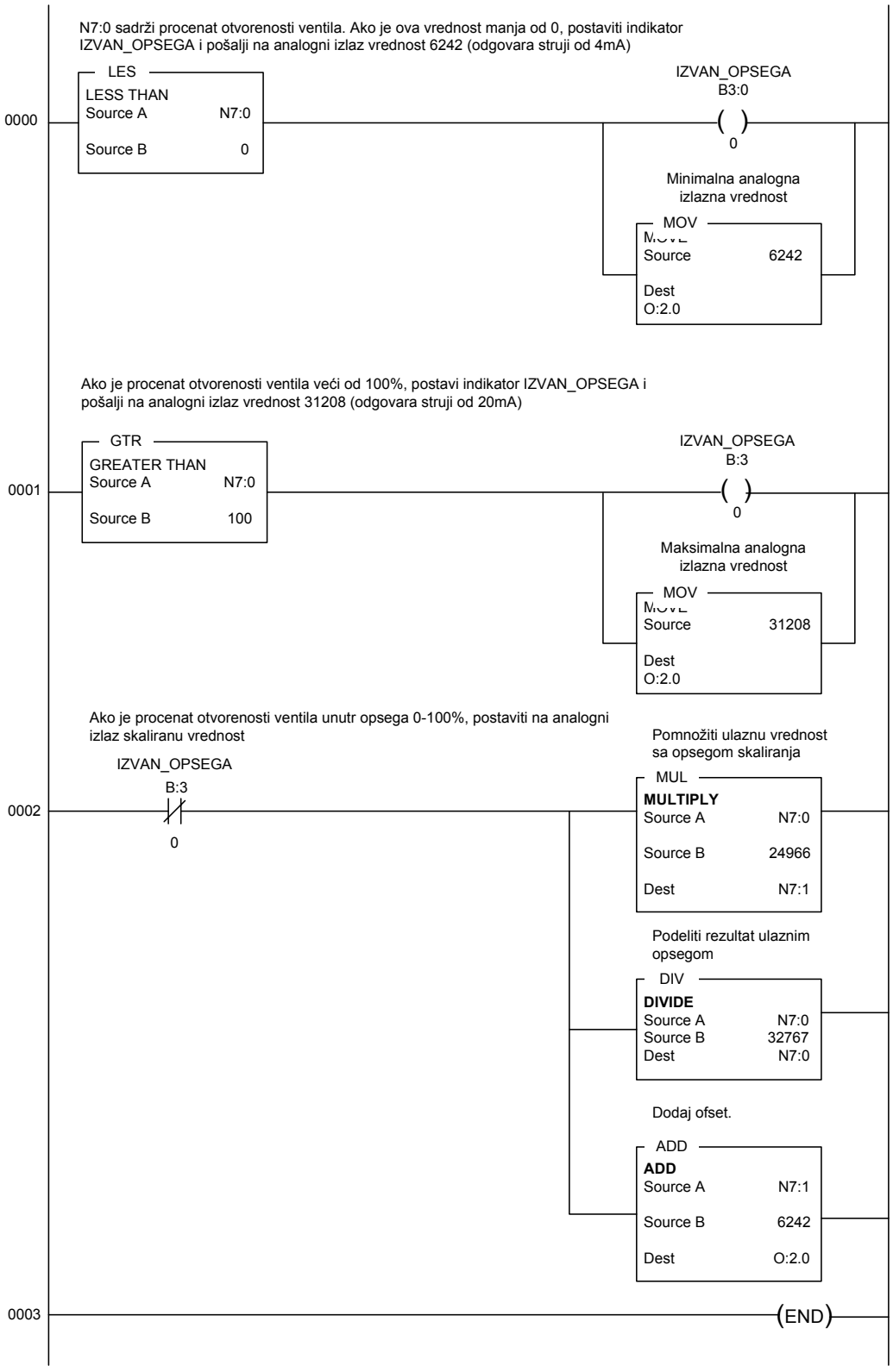
offset = *scaled_min* – (*input_min* x *slope*) = 6242 – (0x(24966/100)) = 6242

Dakle, ako je *input_value* u opsegu dozvoljenih vrednosti (0-100), *scaled_value* (vrednost koja se šalje analognom modulu) određuje se shodno izvednoj jednačini. Ako je *input_value* manje od 0% (podkoračenje), *scaled_value* se postavlja na 6242 (*scaled_min*). Ako je *input_value* veće od 100% (prekoračenje), *scaled_value* se postavlja na 31208 (*scaled_max*).

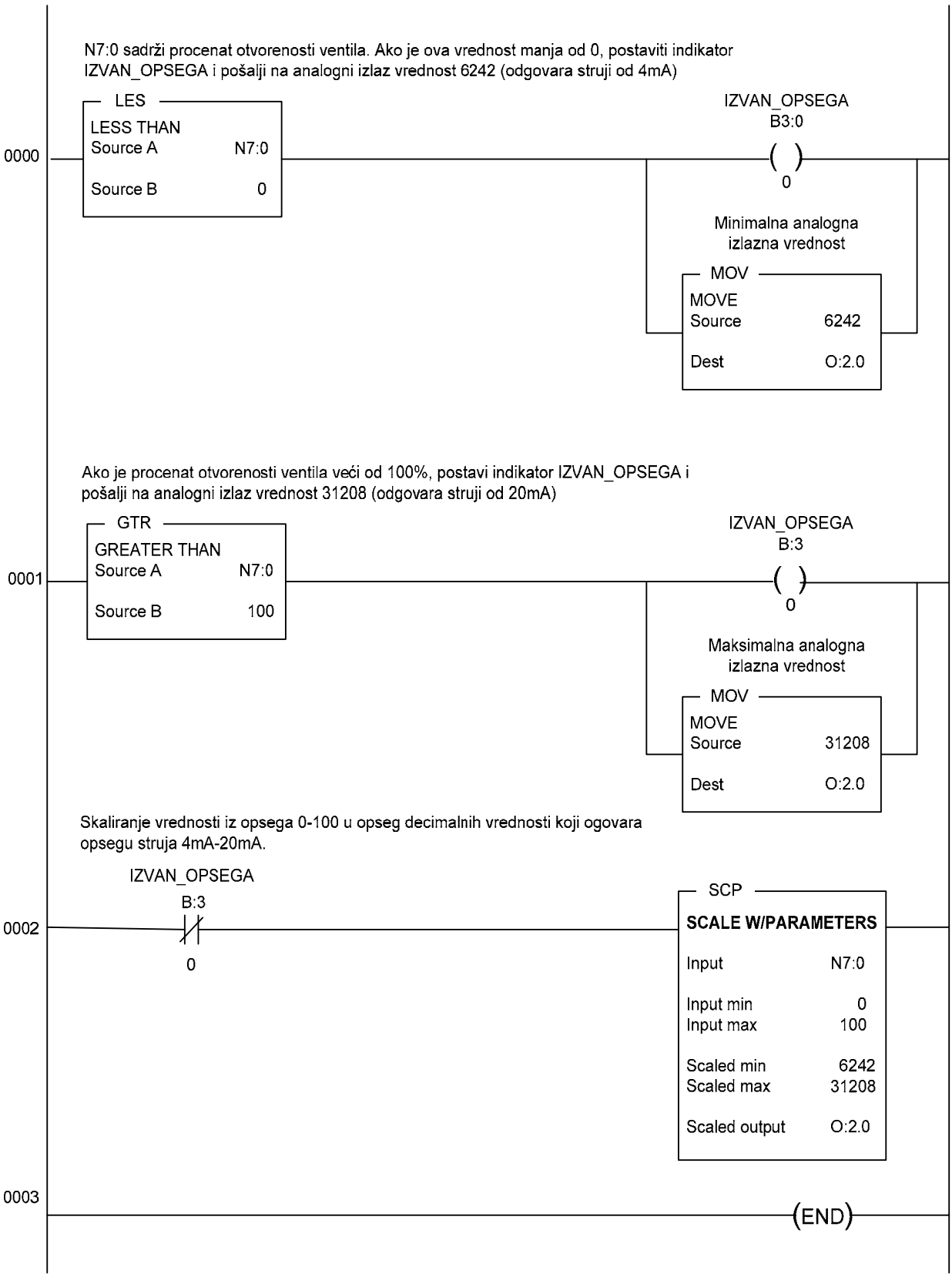


Sl. 4-8 Zavisnost izlazne vrednosti od zadate procentualne otvorenosti ventila.

Treba uočiti da obe vrednosti *slope* i *offet* pripadaju dozvoljenom opsegu celih brojeva (integers), (-32768 - +32767), te je stoga za izračunavanje skalirane izlazne vrednosti moguće direktno primeniti standardne matematičke naredbe. Leder program koji koristi standardne matematičke naredbe i naredbe poređenja prikazan je na Sl. 4-9. Na Sl. 4-10 je prikazan leder program kod koga se za skaliranje izlazne vrednosti koristi SCP naredba.



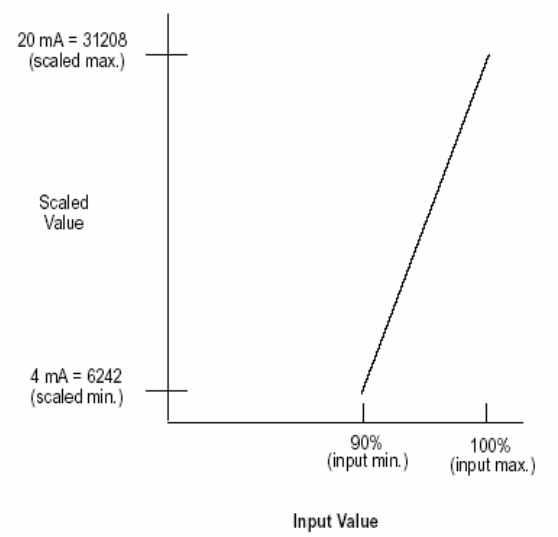
Sl. 4-9 Leder program za pobudu aktuatora ventila (varijanta sa matematičkim naredbama).



Sl. 4-10 Leder program za pobudu aktuatora ventila (varijanta sa SCP naredbom).

Primer: Skaliranje za ofset veći od 32767 ili manji od -32768.

Zadatak: Neka su uslovi iz prethodnog primera promenjeni tako da opsegu struja 4mA-20mA odgovara otvorenost ventila od 90-100%. Analogni modul NIO4I postavljen je u slot 2, a aktuator električnog ventila povezan je na izlazni kanal 0 analognog modula.



Sl. 4-11 Zavisnost skaliranja sa velikom negativnom vrednošću offset-a.

Rešenje: Na Sl. 4-11 prikazana je zavisnost skalirane izlazne vrednosti od ulazne vrednosti. Kao što se može videti, minimalna ulazna vrednost (*input_min*) iznosi 90% i njoj odgovara minimalna skalirana izlazna vrednost *scaled_min*=6242 (za struju od 4mA). Maksimalna ulazna vrednost iznosi *input_max*=100% i njoj odgovara maksimalna izlazna skalirana vrednost *scaled_max*=31208 (za struju od 20mA). Na osnovu ovih podataka možemo odrediti jednačinu skaliranja:

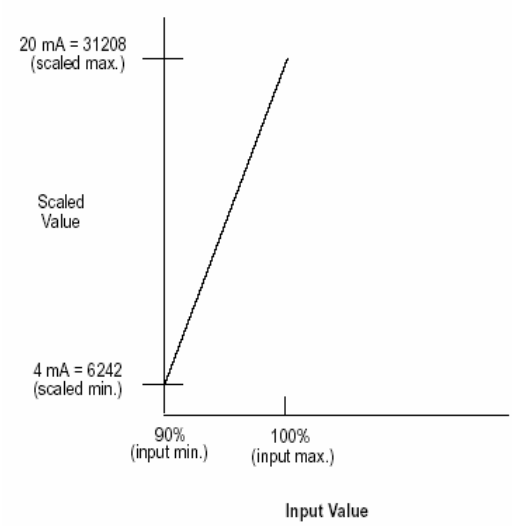
scaled_value = (*input_value* x *slope*) + *offset*, gde je:

slope = (*scaled_max – scaled_min*)/(*input_max – input_min*) =
= (31208 – 6242)/(100-90) = 24966/10

offset = *scaled_min* – (*input_min* x *slope*) = 6242 – (90x(24966/100)) = -218452, odnosno
scaled_value=(*input_value*) x (24966/10) – 218452

Uočimo da je vrednost *offset*-a manja od -32768, odnosno manja od najmanje negativne vrednosti koja se može predstaviti sa 16 bita što onemogućava direktnu primenu matematičkih naredbi.

Da bi se pojednostavilo izračunavanje, linearna zavisnost se može translirati duž ose ulaznih vrednosti ka koordinatnom početku, kao na Sl. 4-12. Ova zavisnost se dobija nakon uvedene smene *input_value* = *input_value* – *input_min*. S obzirom na to, jedančina skaliranja sada glasi:



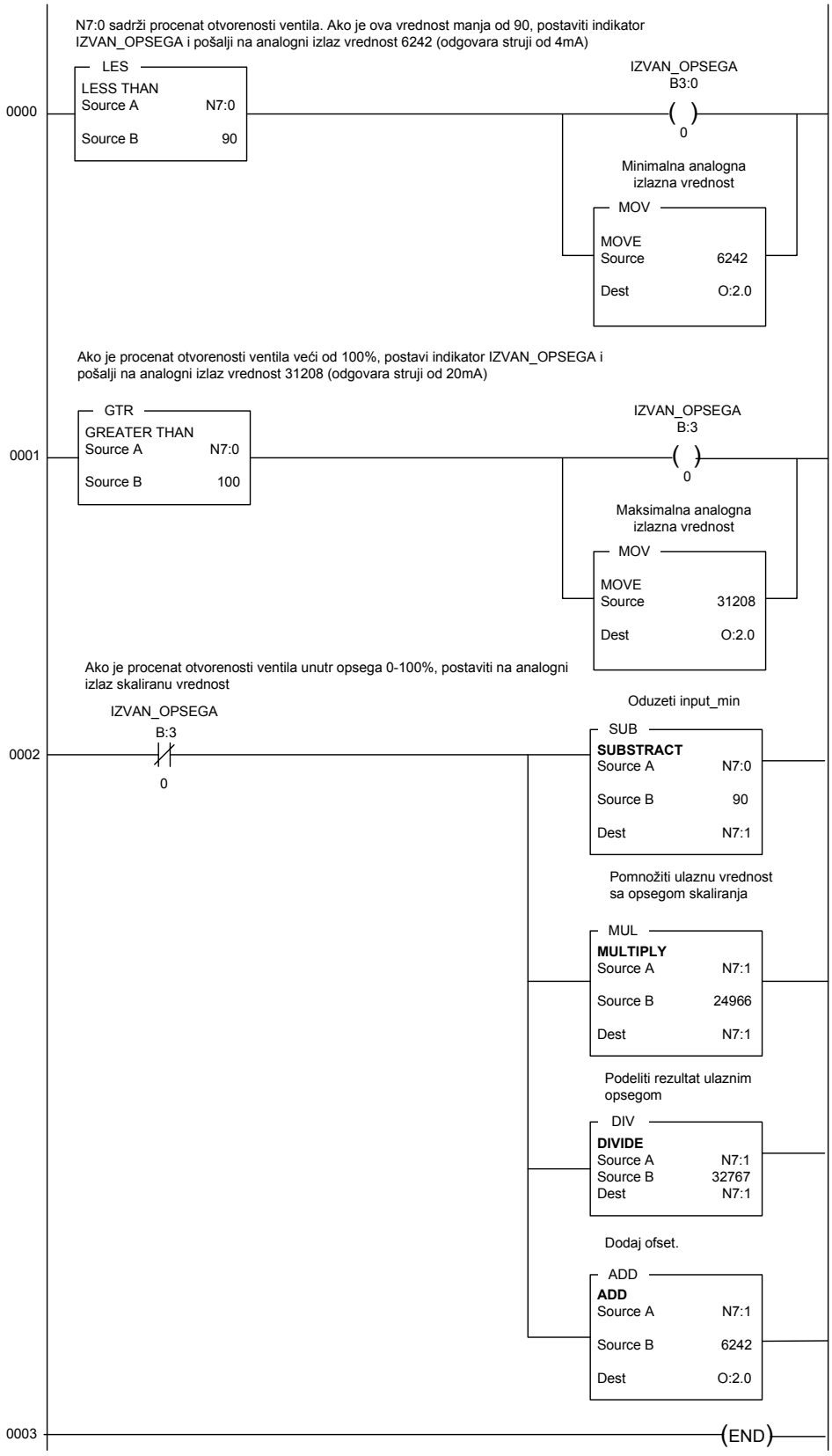
Sl. 4-12 Translirana zavisnost skaliranja.

scaled_value= ((*input_value–input_min*)x *slope*) + *offset*, gde je:

slope=(*scaled_max–*
scaled_min)/(*input_max – input_min*) = (31208 – 6242)/(100-90) = 24966/10

offset = *scaled_min* = 6242
scaled_value=(*input_value* - 90) x (24966/10) + 6242

Očigledno, *offset* je smanjen i može se predstaviti kao integer. Odgovarajući leder program prikazan je na Sl. 4-13. Kao i u prethodnim primerima, umesto standardnih matematičkih naredbi možemo koristi naredbu za skaliranje podatak SCP ili SCL.



Sl. 4-13 Leder program za transliranu zavisnost skaliranja.

Primer:

Zadatak: U slotu 1 modularnog PLC kontrolera smešten je analogni modul NIO4V. Senzor pritiska za opseg 0 – 200 psi (paskala) povezan je na ulazni kanal 0 analognog modula. Senzor pritiska daje struju 4mA-20mA, tako da struja od 4mA odgovara pritisku od 0psi, a struja od 20mA odgovara pritisku od 200psi. Ulaznu vrednost dobijenu od analognog modula treba najpre testirati kako bi se proverilo da li pripada dozvoljenom opsegu, a zatim skalirati na opseg napona 0-2.5V i proslediti voltmetru koji je povezan na izlazni kanal 0 analognog modula. Ako je očitana ulazna vrednost izvan dozvoljenog opsega, treba postaviti indikator IZVAN_OPSEGA.

Rešenje: Opseg pune skale ulaznog kanala analognog modula NIO4V, konfigurisanog kao strujni ulaz, iznosi +/-20mA. Opsegu pune skale odgovara decimalni opseg +/-16384. Na osnovu ovih podataka, lako dolazimo do graničnih vrednosti decimalnog opsega koji odgovara ulaznoj struju od 4mA do 20mA:

input_min = (16384/20mA) x 4mA = 3277

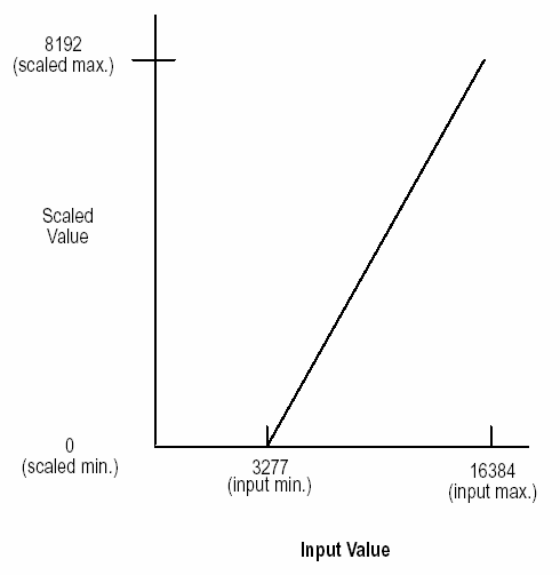
input_max = (16384/20mA) x 20mA = 16384

Izlazni kanal analognog modula NIO4V, na koji je povezan volmetar, je naponskog tipa sa opsegom pune skale +/-10V. Opsegu pune skale odgovara decimalni opseg -32768 - +32764. Na osnovu ovih podataka možemo odrediti decimalni opseg koji odgovara željenom naponskom opsegu od 0-2.5V:

scaled_min = 0

scaled_max = (32764/10V) x 2.5V = 8192

Na Sl. 4-14 je prikazana zavisnost između decimalnih vrednosti ulazne veličine (*input_value*) i rezultujućih skaliranih vrednosti izlazne veličine (*scaled_value*).



Sl. 4-14 Zavisnost skaliranja.

Jednačina skaliranja je oblika:

scaled_value = (*input_value* x *slope*) + *offset*, gde je:

slope = (*scaled_max* – *scaled_min*)/(*input_max* – *input_min*)

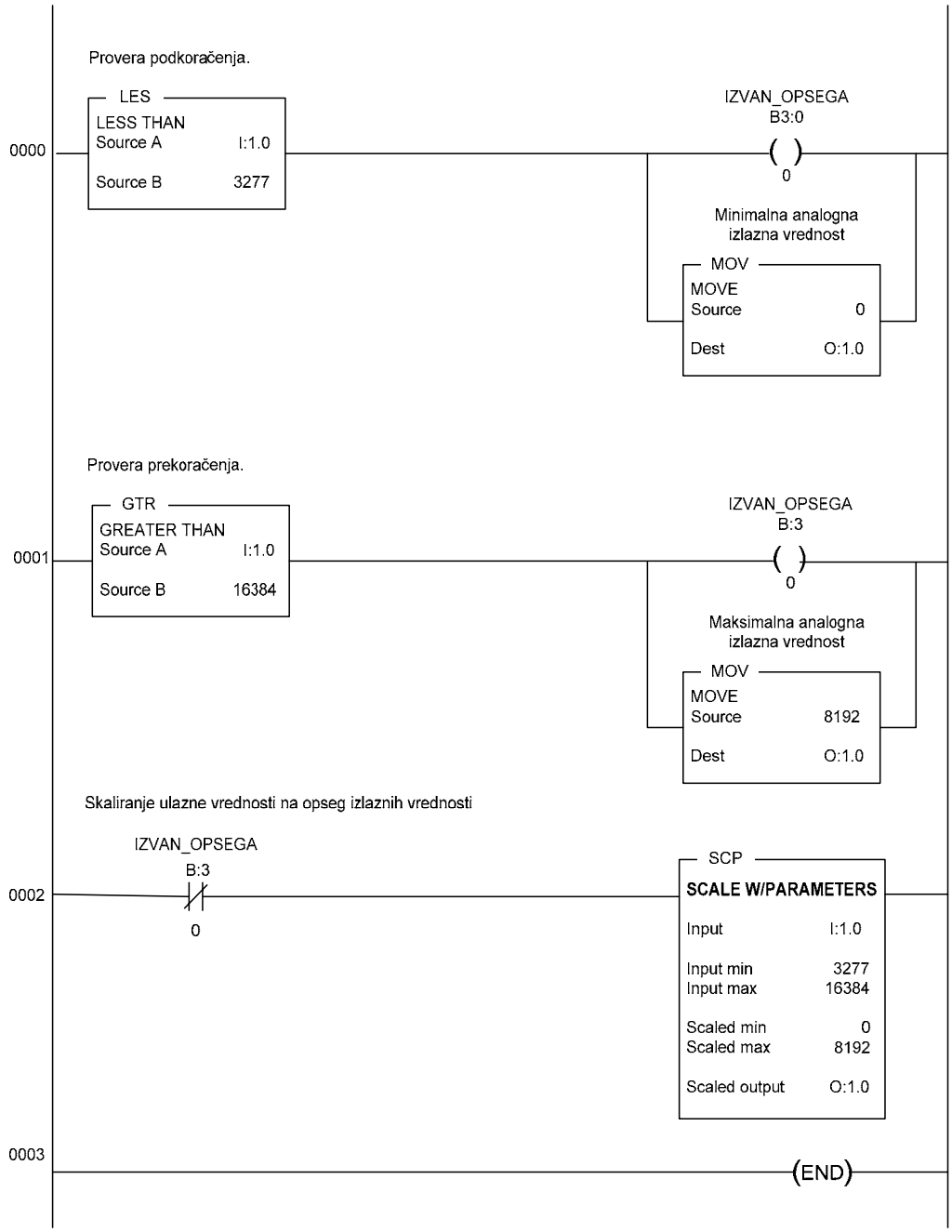
= (8192 – 0)/(16384-3277) = 8192/13170

offset = *scaled_min* – (*input_min* x *slope*)

= 0 – (3277x(8192/13107)) = -2048, odnosno

scaled_value=(*input_value*) x (8192/13017) – 2048

Na Sl. 4-15 je prikazan odgovarajući leder program. Program, najpre ispituje da li očitana ulazna vrednost pripada dozvoljenom opsegu. Ako je ulazna vrednost izvan dozvoljenog opsega, postavlja se idikator IZVAN_OPSEGA. Ako je ulazna vrednost pripada dozvoljenom opsegu, izračunava se izlazna decimalna vrednost i prosleđuje analognom modulu. Skaliranje ulazne vrednosti se na zadati opseg izlaznih vrednosti obavlja se uz pomoć naredbe SCP.

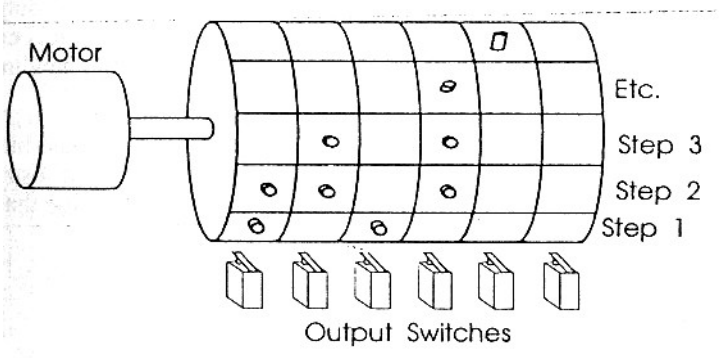


Sl. 4-15 Leder program

4.4 Sekvencijalno upravljanje

Jedan od 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. Zamislimo, na primer, proizvodnu liniju za punjenje boca nekom tečnošću. Proces se odvija tako što se boce najpre stavljaju na liniju, a zatim peru, pune, zatvaraju, proveravaju i konačno pakuju – dakle, niz (sekvenca) uvek istih operacija. Mnogi uređaji za domaćinstvo rade na sekvencijalan način. Mašina za pranje rublja je dobar primer sekvencijalnog upravljanja. Bitno obeležje sekvencijalnog 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.

U prošlosti, za realizaciju sekvencijalnog upravljanja korišćeni su doboš-programatori. Doboš je oblika cilindra sa udubljenim rupama, koje su ravnomerno raspoređene duž većeg broja paralelnih, kružnih staza. U svaku rupu se može postaviti trn, a raspored trnova duž jedne staze odgovara binarnoj sekvenci za pobudu jednog izvršnog organa (Sl. 4-16). Neposredno uz doboš postavljen je niz prekidača, pri čemu svaki prekidač upravlja jednim izvršnim organom. Kako se doboš okreće, tako trnovi zatvaraju prekidače na čijim se izlazima generiše željena sekvenca binarno kodiranih upavljačkih reči. Doboš okreće motor, čija brzina rotiranja može da se reguliše. Doboš-programator se lako programira. Korisnik najpre formira tabelu koja za svaki korak (fazu procesa), pokazuje koji izvršni organi su aktivni u tom koraku (Sl. 4-17). Međutim, doboš-programator ima i niz nedostataka. Na primer, iako brzina rotiranja doboša, a time i brzina odvijanja procesa može da se reguliše uz pomoć motora, ne postoji mogućnost programiranja trajanja pojedinih koraka – svi koraci imaju isto trajanje. Takođe, doboš-programator nije u stanju da ispituje ulazne informacije o tekućem stanju procesa i da na osnovu tih informacija, eventualno promeni radnu sekvencu.



Sl. 4-16 Doboš-programer. (*Output Switches – izlazni prekidači; Step 3-1 – koraci 3-1, svaki red trnova odgovara jednom koraku, tj. fazi*)

Step	Input Pump	Heater	Add Cleaner	Sprayer	Output Pump	Blower
1	on		on			
2	on	on		on		
3		on		on		
4					on	
5						on

Sl. 4-17 Primer izlazne sekvence doboš-programatora sa Sl. 4-16. (*Input Pump – ulazna pumpa; Heater – grejač; Add Cleaner – dodaj sredstvo za čišćenje; Sprayer – prskalica; Output Pump – izlazna pumpa*)

Mnoga ograničenja karakteristična za doboš-programatior mogu se prevazići ako se za sekvencijalno upravljanje koristi PLC kontroler. Za razliku od doboš-programatora, kod koga svi koraci imaju isto trajanje, kod PLC realizacije, s obzirom na daleko veću fleksibilnost programiranja, uobičajeno se koristi pristup kod koga se je prelazak sa jedne na drugu aktivnost 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. Dakle, moguće je da se svi parametri koji učestvuju u poređenju, urede u jedan niz binarno kodiranih reči, i da se stanje procesa, koje se učitava preko digitalnih ulaznih modula poredi sa odgovarajućom reči iz niza.

Nema nikakve sumnje da bi se opisane operacije mogle izvesti kombinovanjem naredbi za unošenje i iznošenje digitalnih podataka i naredbi za poređenje. Međutim, pošto je potreba za ovim operacijama izuzetno izražena, predviđene su dve posebne naredbe kojima se one u celosti mogu realizovati. Obe naredbe su *naredbe akcije.*

4.4.1 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.

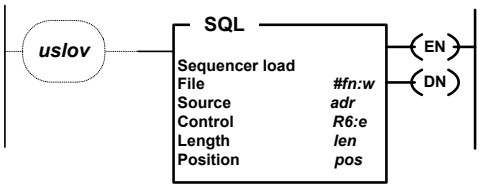
Ako neka aplikacija zahteva da se sekvencijalna obrada izvrši nad podacima koji su duži od 16 bitova, onda se ti podaci moraju podeliti na više datoteka. Tada se u svakom rangu na izlazu kao naredbe *akcije* mogu paralelno staviti više istih naredbi kojima se adresiraju sve definisane datoteke.

Svakoj datoteci koja se specificira u okviru neke sekvencijalne naredbe pridružuje se po jedan elemenat upravljačke datoteke *R*. U okviru ovog elementa pamte se indikatorski bitovi, kao i vrednost pointera i dužina same datoteke. O formatu jednog elementa ove datoteke biće kasnije više reči.

Potrebno je da se naglasi da se ove naredbe ne izvode uvek na isti način. Naime, samo kada se *uslov* u rangu menja sa *neistinit* na *istinit* menja se vrednost pointera i on ukazuje na drugi podatak. Međutim, ako *uslov* posle toga ostane i dalje *istinit*, pointer ne menja vrednost već se naredba izvršava sa podatkom koji je uzet pri poslednjoj promeni pointera.

- **SQL – Sequencer Load (sekvencijalno punjenje datoteke)**

SQO naredba, grafički simbol i položaj urangu



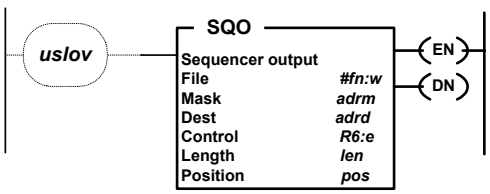
File - bazna adresa odredišne datoteke
Source - adresa izvora podataka (promenljiva, datoteka ili konstanta)
Control - adresa upravljačkog elementa
Length - dužina niza (konstanta <256)
Position - početna pozicija pointera (konstanta) (u toku rada trenutna pozicija pointera)

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. Na taj način se pri svakom sledećem izvršavanju naredbe menja sadržaj sledeće reči u nizu. Ukoliko se kao *source* adresa navede konstanta onda se ceo niz postavlja na istu vrednost. Ako je *source* adresa promenljiva (*fn:s*), onda svaka reč niza dobija vrednost koju promenljiva ima u trenutku izvođenja naredbe. Međutim, ako se kao *source* adresa navede datoteka (*#fn:s*), onda se ta adresa uzima kao *bazna adresa izvorne datoteke*, što znači da se pri izvođenju naredbe podatak uzima sa one adrese na koju u izvornoj datoteci pokazuje pointer. Pri tome se podrazumeva da obe datoteke imaju istu dužinu, definisanu kao *length*.

Pri sledećim sken ciklusima, za svo vreme za koje *uslov* ostaje *istinit*, vrednost pointera se ne menja, već se isti, prethodno određeni, podatak prenosi u promenljivu označenu sa *dest*.

- **SQO – Sequencer output (sekvencijalno upravljanje)**

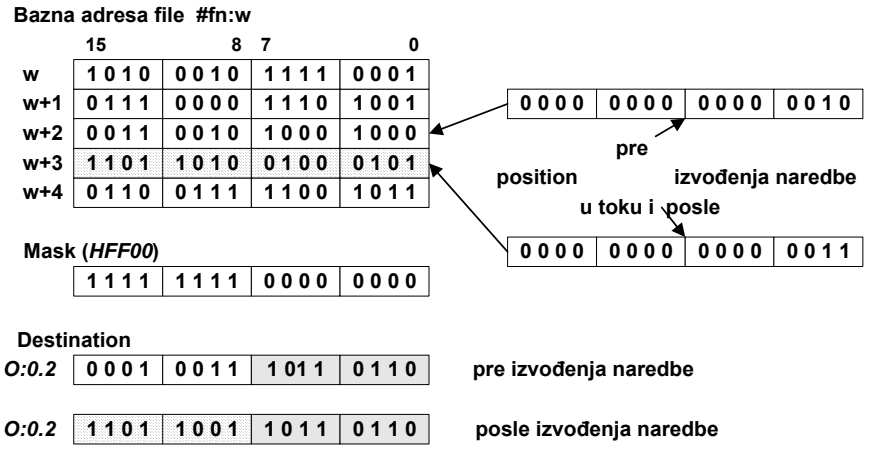
SQO naredba, grafički simbol i položaj urangu



File - bazna adresa izvorne datoteke
Mask - adresa maske promenljiva, datoteka ili heksadecimalna konstanta)
Dest - adresa odredišta (izlazni modul ili datoteka)
Control - adresa upravljačkog elementa
Length - dužina niza (konstanta <256)
Position - početna pozicija pointera (konstanta) (u toku rada trenutna pozicija pointera)

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. Ta reč se filtrira kroz masku *mask* i rezultat filtracije se prenosi u promenljivu označenu sa *dest*. Ako je kao *dest* navedena datoteka *#fn:d* onda će se rezultat upisati u onu reč te datoteke na koju pokazuje pointer (Sl. 4-18). Isto tako, ako je kao *mask* navedena datoteka *#fn:m* onda i maska prestaje da bude fiksna, već se svaki put kao maska uzima ona reč iz datoteke na koju pokazuje pointer. Potrebno je zapaziti da se u reči koja označena sa *dest* menjaju samo oni bitovi koji su nemaskirani (odgovorajući bitovi maske su postavljeni na 1).

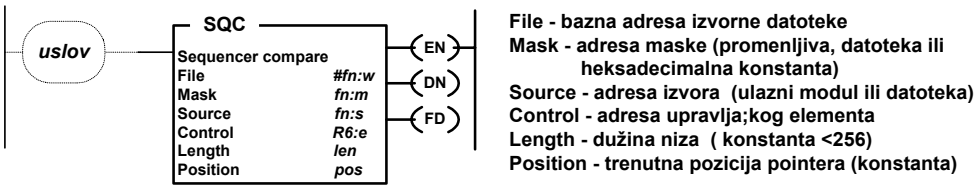
Pri sledećim sken ciklusima, za svo vreme za koje *uslov* ostaje *istinit*, vrednost pointera se ne menja, već se isti, prethodno određeni, podatak prenosi u promenljivu označenu sa *dest*.



Sl. 4-18 Ilustracija izvršavanja SQO naredbe.

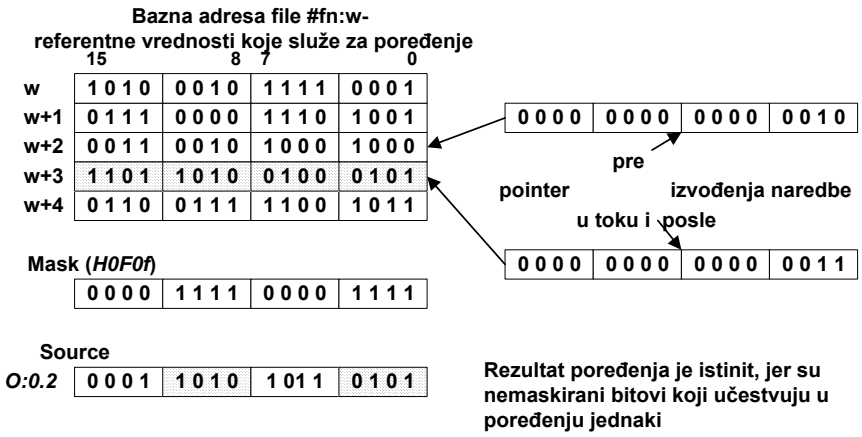
- **SQC – Sequencer compare (sekvencijalno poređenje)**

SQC naredba, grafički simbol i položaj urangu



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. Ta reč se poredi sa filtriranim podatkom koji sadrži promenljiva označena kao *source* i rezultat poređenja se upisuje u odgovarajući indikatorski bit. Filtracija podatka vrši se pomoću maske *mask* i to tako da u poređenju učestvuju samo oni bitovi kojima u maski odgovara vrednost bita 1 (nemaskirani bitovi). Ako je kao *source* navedena datoteka *#fn:s* onda će se podaci koji učestvuju u poređenju uzimati iz one reči te datoteke na koju pokazuje pointer (Sl. 4-19). Isto tako, ako je kao *mask* navedena datoteka *#fn:m* onda i maska prestaje da bude fiksna, već se svaki put kao maska uzima ona reč iz datoteke na koju pokazuje pointer.

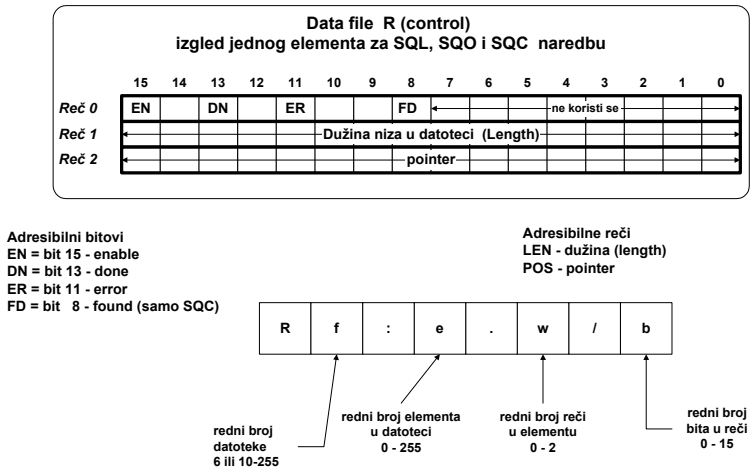
Pri sledećim sken ciklusima, za svo vreme za koje *uslov* ostaje *istinit*, vrednost pointera se ne menja, već se isti, prethodno određeni, podatak (*source*) uzima kao podatak za poređenje.



Sl. 4-19 Ilustracija izvršavanja SQC naredbe.

Datoteka R – Control

Naredbama za sekvenciranje pridružuju se indikatorski bitovi i upravljački parametri. Ove informacije se smeštaju u *upravljačku datoteku* tipa *R*. Pri tome se može koristiti sistemska upravljačka datoteka broj *6*, ili korisnička datoteka (brojevi od 9 do 255). Jedan elemenat ove datoteke, koji se odnosi na SQO i SQC naredbu ima izgled kao na Sl. 4-20.



Sl. 4-20 Elemenat upravljačke datoteke za SQO i SQC naredbu.

Bitovi stanja u elementu datoteke *R* menjaju se na sledeći način:

EN – Enable bit se postavlja na 1 kada *uslov* prelazi sa *neistinit* na *istinit*. Postavljanje ovog bita prouzrokuje da se izvrši naredba i vrednost pointera poveća za 1. Pri svakom sledećem prolazu kroz ovaj rang, sve dok je uslov *istinit*, EN bit ozadržava vrednost 1, ali se vrednost pointera ne menja, već se naredba izvršava sa istom vrednošću pointera. Kada *uslov* postane *neistinit*, EN – bit se resetuje na 0.

DN – Done bit se postavlja na 1 kada vrednost pointera, posle niza izvođenja SQL, SQO ili SQC naredbe, dođe do kraja niza u zadanoj datoteci. Ovaj bit će biti resetovan na 0 tek u onom sken ciklusu u kome *uslov*, pošto je prethodno postao *neistinit*, ponovo postaje *istinit* (kada se EN-bit ponovo postavi na 1).

ER – Error bit se postavlja na 1 kada se u programu detektuje negativna vrednost pointera, ili negativna ili nulta vrednost dužine niza. Ako se ovaj bit ne resetuje pre kraja sken ciklusa nastaće *značajna greška*.

FD – Found bit se postavlja na 1 ako je rezultat poređenja u SQC naredbi *istinit*. Drugim rečima ovaj bit ukazuje na to da su nemaskirani bitovi podataka jednaki odgovarajućim bitovima u datoteci referentnih vrednosti.

Length i position

Promenljive *length* se pamti u prvoj reči datoteke R i predstavlja broj reči koji se nalazi u nizu u jednoj sekvencijalnoj datoteci. Maksimalna vrednost dužine je 255. Pri definisanju dužine, potrebno je voditi računa o činjenici da navedena adresa reči *w* u datoteci *#fn:w* zapravo predstavlja nultu, početnu poziciju. To znači da se za datu dužinu *l* u datoteci koristi zapravo *l*+1 reč. Ovo se naravno odnosi i na *mask, source* i *dest* ukoliko su u naredbi specificirane kao datoteke.

Vrednost pointera, označena kao *position*, pamti se u drugoj reči datoteke R. Početna vrednost pointera se definiše pri specifikaciji naredbe. Vrednost pointera se kreće od 1 do *l* i ukazuje na reči u datoteci od *fn:*(*w*+1) do *fn:*(*w*+*l*+1). Kada pointer stigne do poslednje reči u datoteci, postavlja se DN-bit na 1 i pri tome se u prvom sledećem sken ciklusu u kome *uslov* ima prelaz sa *neistinit* na *istinitinit* (isiti ciklus u kome se resetuje DN-bit) vrednost pointera automatski vraća na 1. Pri definiciji početne vrednosti pointera potrebno je obratiti pažnju na činjenicu da se ona poveća za 1 pre prvog izvođenja naredbe.

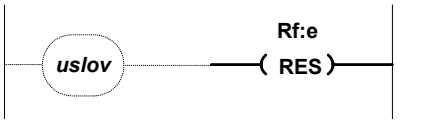
Ako se kao početna vrednost pointera definiše 0, onda će pri izvođenju SQC naredbe obrada početi od reči u datoteci čija je adresa *fn:*(*w*+1). Međutim kod SQO naredbe način izvođenja operacije zavisi od istinitosti *uslova* u prvom sken ciklusu. Ako je *uslov istinit* naredba se izvršava počev od nulte reči, čija je adresa *fn:*(*w*+0). Međutim, ako je uslov *neistinit*, izvršavanje naredbe se odlaže sve dok uslov ne postane *istinit* i tada se uzima prva reč, čija je adresa.

Konačno, važno je da se istakne da se prilikom eventualne programske promene dužine i pozicije mora voditi računa da se ne prekorači veličina definisanog niza u datoteci.

Resetovanje parametara

Ukoliko se iz nekog razloga želi prekinuti sekvencijalno upravljanje ili poređenje, to se može ostvariti pomoću *RES* naredbe u kojoj se navodi adresa nulte reči elementa datoteke R koji je vezan za naredbu čiji se rad želi resetovati *Rn*:*e*. *RES* naredbom se vrednosti svih indikatorskih bitova, izuzev *FD*-bita, postavljaju na 0. Istovremeno se i vrednost pointera postavlja na 0 (ova vrednost će se povećati na 1 pre prvog sledećeg izvođenja naredbe).

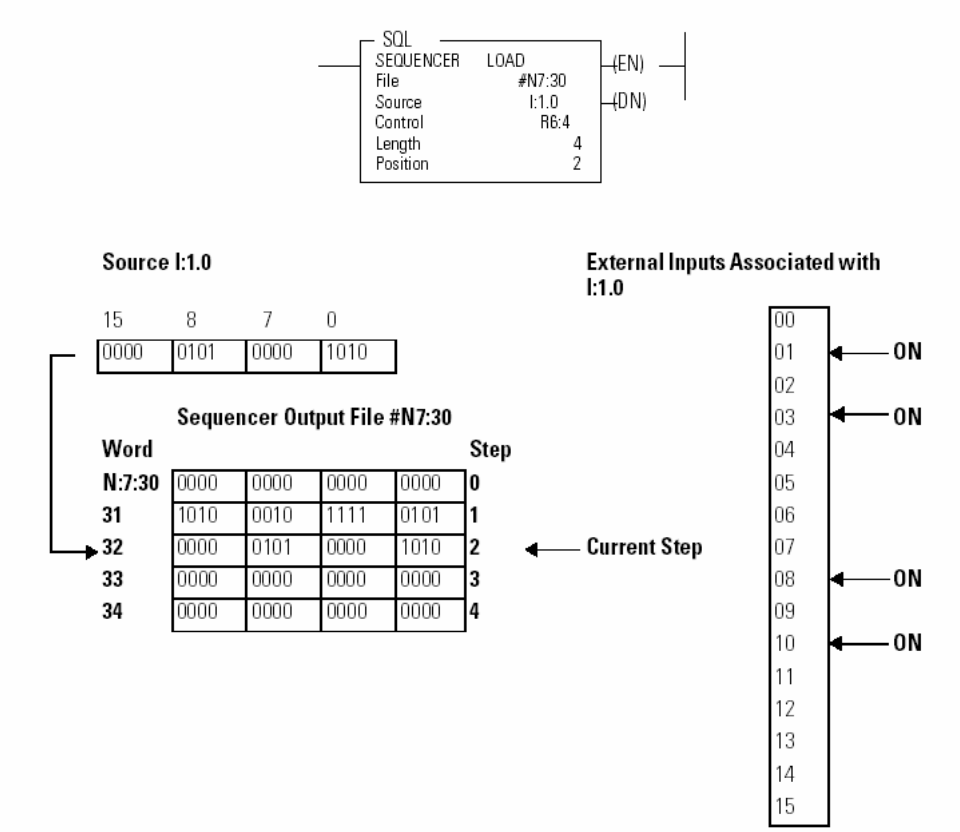
RES naredba, grafički simbol i položaj u rangu



4.4.1.1 Korišćenje SQL naredbe

Na Sl. 4-21 je prikazan primer SQL naredbe. Paremetri SQL naredbe su tako podešeni da je ulazna reč I:1.0 izvor podataka, a da se pročitani podaci smeštaju u odredišnu datoteku #N7:30 veličine 4 reči. Kada se uslov ranga promeni sa *neistina* na *istina*, pointer SQL naredbe se uvećava za jedan i podatak očitan sa ulaza I:1.0 se upisuje na poziciju u

odredišnom fajlu na koju ukazuje pointer. SQL naredba nastavlja da prenosi podatke sa ulaza na istu poziciju odredišne datoteke sve dok je uslov ranga istinit. Nakon što je korak 4 završen, setuje se bit DN. Sledeća promena uslova sa neistina na istina, vraća pointer na poziciju 1 (element N7:31). Da je umesto ulazne reči I:1.0, kao izvor navedena interna datoteka, npr. #N7:40, tada bi obe datoteka odredišna i izvršna bile iste dužine (4), a SQL nardba bi, praktično, kopirala sadržaj izvorene datoteke u odredišnu.



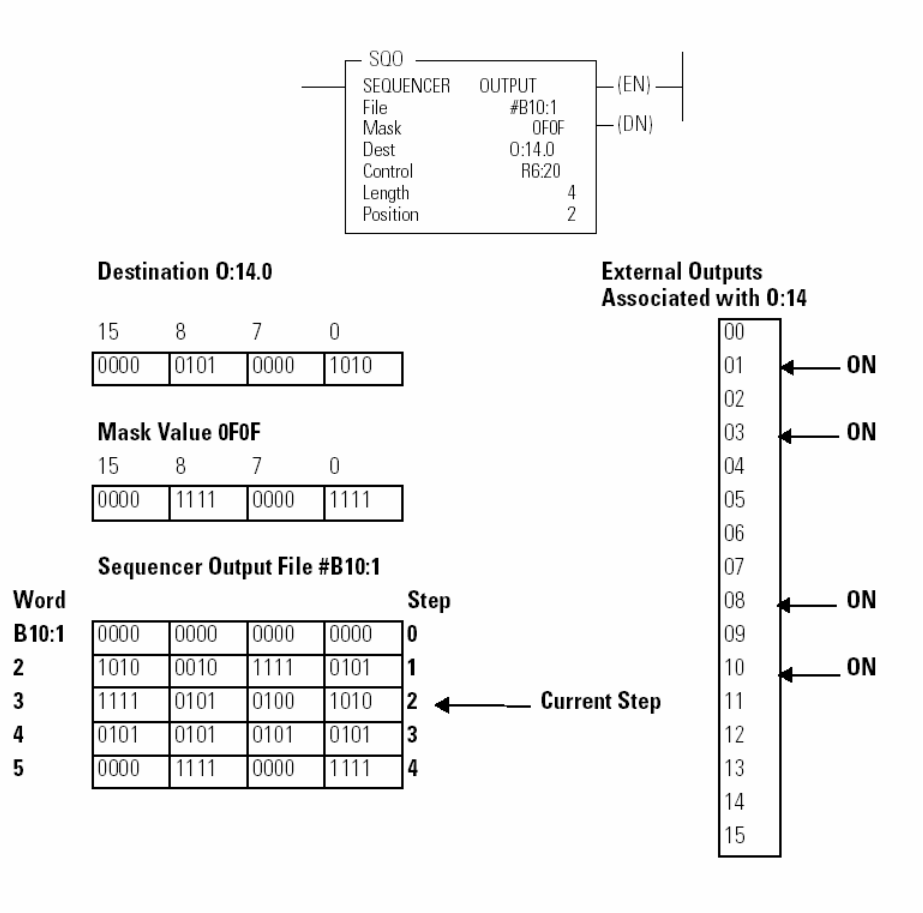
Sl. 4-21 Primer SQL naredbe.

4.4.1.2 Korišćenje SQO naredbe

SQO naredba čita reči specificiranog fajla čiji su bitovi namenjeni upravljanju izvršnih organa. Kada uslov ranga koji sadrži SQO naredbu postane istinit, naredba čita sledeću reč iz fajla. Pročitana reč prolazi kroz masku i prenosi se na odredišnu adresu. Samo oni bitovi pročitane reči kojima u maski odgovara bit 1 se prenose. Bit DONE postaje 1 nakon što je iz fajla pročitana poslednja reč. Sledeći promena uslova ranga sa *neistina* na *istina* usloviće da SQO naredba krene iz početka, sa pointerom koji je resetovan na početnu vrednost 1.

Na Sl. 4-22 je ilustrovan rad SQO naredbe. SQO naredba sa slike definiše sekvencu od 5 reči smeštenu u datoteci B10, počev od reči B10:1. Početna vrednost pointera je 2, što znači da će kao prva reč iz izvorišnog fajla biti pročitana reč B10:3. Odredišna adresa je prva reč izlazne datoteke O:14. (Dugim rečima, odredište pročitanih reči iz ulaznog fajla je izlazni digitalni modul sa adresom 14.) Naredba SQO definiše masku čija je heksadecimalna vrednost 0F0F. To znači da se na odredište ne prenose svi bitovi pročitanih reči, već samo osam bita, naznačenih jedinicama u maski (to su bitovi 0-3 i bitovi 8-11). Naravno, sadržaj izvorišnog fajla mora biti unapred pripremljen, prema zahtevima aplikacije, i upisan u

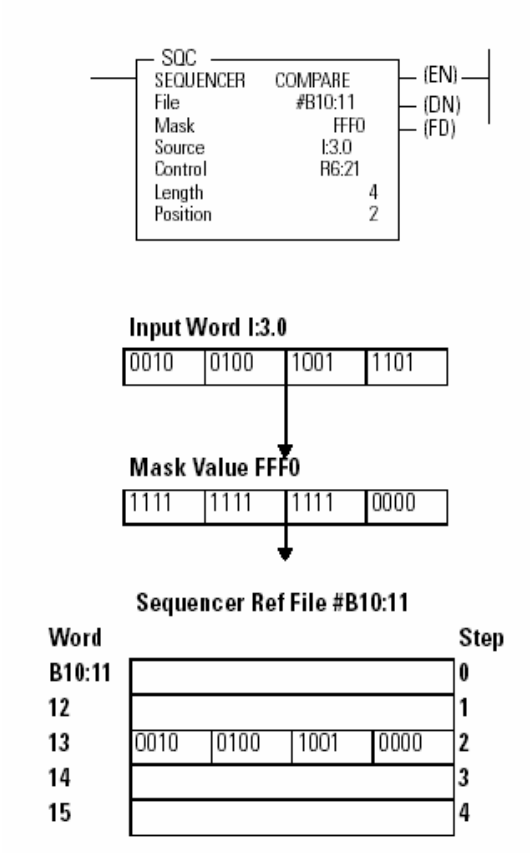
odgovarajuće reči naznačene datoteke. Očigledno, SQO naredba predstavlja softverski ekvivalent doboš-programatora.



Sl. 4-22 Primer SQO naredbe.

4.4.1.3 Korišćenje SQC naredbe

SQC naredba radi tako što poredi sve ne-maskirane bitove iz izvorišne reči sa odgovarajućim bitovima referencirane reči. Ako na svim bitskim prozicijama postoji slaganje, naredba SQC postavlja bit FD kontrolne reči na 1. U suprotnom, ako postoji nesleganje barem na jednoj poziciji, bit FD se postavlja na 0. Na Sl. 4-23 je prikazan primer SQC naredbe. Ova SQC naredba definiše referentni fajl dužine 5 reči šmešten u datoteci B10, počev od reči 11 ove datoteke. Izvorišna adresa (I:3.0) ukazuje na na prvu reč ulazne datoteke I3 (odgovara ulaznom digitalnom modulu sa adresom 3). Tekuća pozicija pointera je 2, što znači da se vrednost pročitana sa ulaza I:3.0 poredi sa rečju B10:13. Ne porede se celokupne reči već samo viših 12 bita, što je definisano maskom 0FFF. Pošto su u konkretnom primeru svi odgovarajući ne-maskirani bitovi identični, SQC naredba setuje FD bit, tj. bit R6:21/FD.



Sl. 4-23 Primer SQC naredbe.

4.4.2 Primeri

Primer: Primena SQO instrukcije.

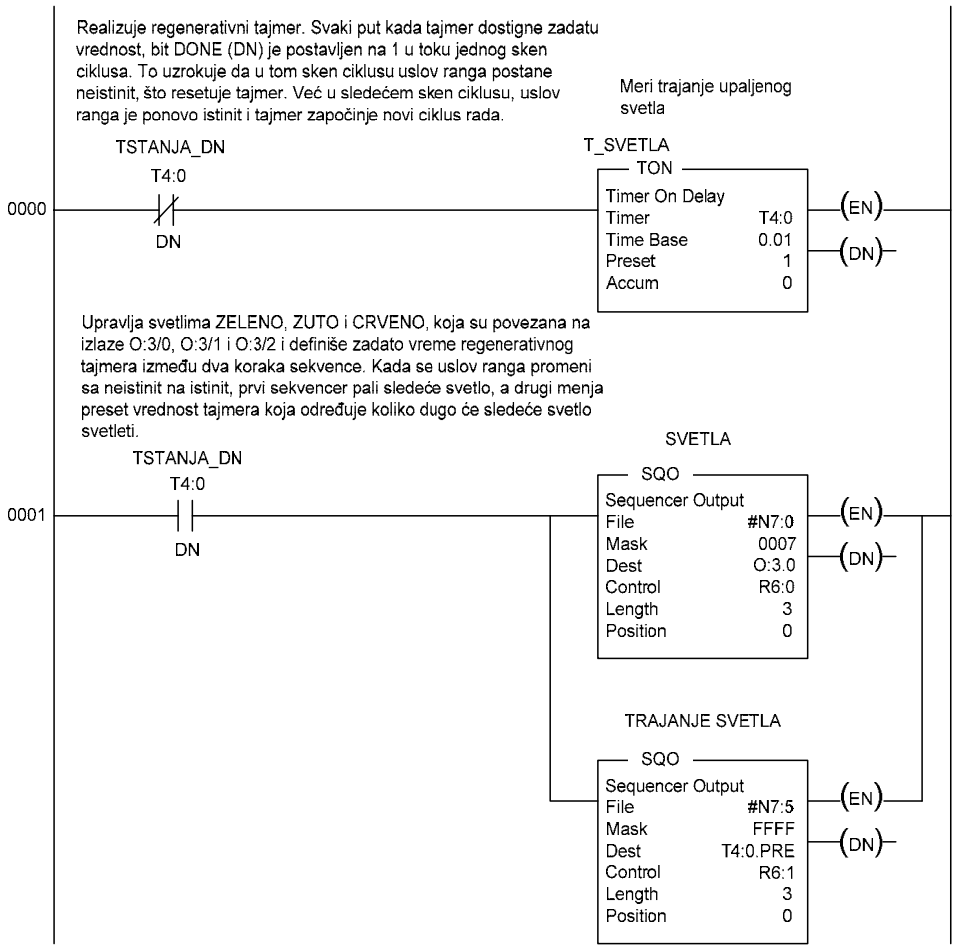
Zadatak: Realizovati leder program za upravljanje semaforom. Semafor ima tri sijalice: ZELENO, ZUTO i CRVENO, koje naizmenično svetle u trajanju od:

ZELENO	60s
ZUTO	15s
CRVENO	30s

Rešenje: Usvajamo da su semaforska svetla povezana na izlazni modul na sledeći način:

ZELENO	O:3/0
ZUTO	O:3/1
CRVENO	O:3/2

Leder program je prikazan na Sl. 4-24. Leder program koristi TON naredbu za merenje vremena i SQO naredba za realizaciju sekvencijalnog upravljanja. Uvek kada tajmer dostigne zadatu vrednost, izvršavaju se dve SQO naredbe. Rang 0 realizuje regenerativni tajmer. Po dostizanju zadate vrednosti, bit DONE (DN) tajmera postaje 1. S obzirom da se bit DN koristi kao uslov u rangu 1, dostizanje zadate vrednosti tajmera inicira izvršenje dve SQO instrukcije. Prva SQO naredba čita sledeću reč iz svog izvorišnog fajla i prenosi je na izlazni modul koji pobuđuje sijalice. Sadržaj izvornog fajla prve SQO naredbe prikazan je na Sl. 4-25(a). Druga SQO naredba čita sledeću reč iz svog izvorišnog fajla i upisuje je u PRE registar tajmera, čime se definiše vreme u toku koga će upravo upaljeno svetlo ostati upaljeno. Sadržaj izvornog fajla druge SQO naredbe prikazan je na Sl. 4-25(b).



Sl. 4-24 Leder dijagram kontrolera jednostavnog semafora.

Adresa	Podatak (binarno)			
	15-3	2	1	0
N7:1	0…0	0	0	1
N7:2	0…0	0	1	0
N7:3	0…0	0	0	1

(a)

Adresa	Podatak (decimalno)
N7:6	6000
N7:7	1500
N7:8	3000

(b)

Sl. 4-25 (a) sadržaj izvorišne datoteke sekvencera SVETLA; (b) sadrzaj izvorne datoteke sekvencera TRAJANJE SVETLA.

Primer: Realizacija sekvence

Zadatak: Dat je sistem koji sadrži jedan jednosmerni solenoid (A) i dva dvosmerna (B i C). Potrebno je realizovati sledeću sekvencu pomeranja klipova: A+ B+ C+ B- A- C- . Pri tome se predpostavlja da su u početnom trenutku svi klipovi uvučeni. Granični prekidači koji indiciraju uvučenost klipa A i B su *normalno zatvoreni*, dok su svi ostali granični prekidači *normalno otvoreni*. Sistem se pušta u rad pomoću pritiska na taster i prestaje sa radom kada se jedanput izvrši zahtevana sekvenca.

Rešenje: Postavljeni zadatak je već jednom rešen, ali bez korišćenja naredbi za sekvenciranje. Ovom prilikom zadatak će biti rešen korišćenjem naredbi SQO i SQC. Sistem ima 7 ulaza, SWA-, SWA+, SWB-, SWB+, SWC-, SWC+ i START i 5 izlaza, MOVE_A+, MOVE_B+, MOVE_B-, MOVE_C+ i MOVE_C-. U početnom stanju (korak 0), klipovi sva tri solenoida su uvučeni, a da bi se otpočelo sa radom čeka se pritisak na prekidač START. Kada se pritisne prekidač START, sistem prelazi na korak 1. U ovom koraku, aktivan je izlaz

MOVE_A+ (klip solenoida A se izvlači) i čeka se da klip dođe u krajnji izvučeni položaj (SWA+ = 1). U koraku 2, koji počinje kada klip solenoida A dostigne krajnji izvučeni položaj, obavlja se izvlačenje klipa solenoida B: MOVE_B+ = 1 i čeka se na SWB+ = 1. Pri tome, zadržava se pobuda solenoida A kako bi klip solenoida A ostao u krajnjem izvučenom položaju (tj. MOVE_A+ = 1). Dakle, sistem prolazi kroz sekvencu koraka pri čemu su za svaki korak definisani aktivni izlazi kao i stanje ulaza na koje se čeka kako bi sistem prešao na sledeći korak. Za postavljanje izlaza biće korišćena naredba SQO, a za ispitivanje uslova za prelaz na sledeći korak naredba SQC. Izvorne datoteke obe naredbe imaće istu dužinu, jednaku broju koraka sekvence uvećanom za 1. Ove dve naredbe rade u paru, sinhronizovano, tako da pointeri u izvornim datotekama obe naredbe ukazuju na iste pozicije. Par reči sa istih pozicija dve izvorne datoteke odgovara jednom koraku. Reč iz izvorne datoteke naredbe SQO definiše stanje izlaza, a reč iz izvorne datoteke naredbe SQC na uslov prelaska na novi korak. Najjednostavni način za sihnhronizaciju dve naredbe je da one dele isti kontrolni registar. Kao što se vidi sa Sl. 4-20, kontrolni registar naredbe sekvenciranja pored indikatorskih bitova sadrži i tekuću poziciju pointera na izvornu datoteku. Ako dve naredbe dele isti kontrolni registar, onda će one deliti zajednički pointer, tj. uvećanje pointera u okviru jedne naredbe značiti i uvećanje pointera one druge naredbe. Pri tome, leder dijagram će biti tako konstruisan da je naredba SQC uvek aktivna, a da se naredba SQO aktivira samo kada je indikator FD postavljen na 1, tj. kada očitani ulaz postane jednak očekivanoj vrednosti. S obzirom da naredba SQO/SQC obavlja inkrementiranje pointera uvek kada se uslov ranga promeni sa *neistinit* na *istinit*, inkrementiranje pointera obavljaće samo naredba SQO. Leder program je prikazan na Sl. 4-26. Rang 0 resetuje sekvencer u prvom sken ciklusu. Rang 1 sadrži SQC naredbu koja je uvek aktivna (izvršava se u svakom sken ciklusu). Rang 2 sadrži SQO naredbu koja se izvršava samo u onim sken ciklusima kada je FD=1, tj. kada treba preći na sledeći korak. Sadržaj izvornih datoteka naredbi SQC i SQO definisan je tabelama T. 4, T. 5 i T. 6. Datoteka maski naredbe SQC definiše bitove ulazne reči koji će biti poređeni sa bitovima odgovarajuće reči iz ulazne datoteke podataka. Na primer, u prvom koraku, od interesa je samo stanje prekidača START. Zato maske za prvi korak ima vrednost 1 na poziciji bita 6 (odgovara ulazu START) i sve nule na ostalim pozicijama.

T. 4 Sadržaj datoteke podataka za poređenje naredbe SQC

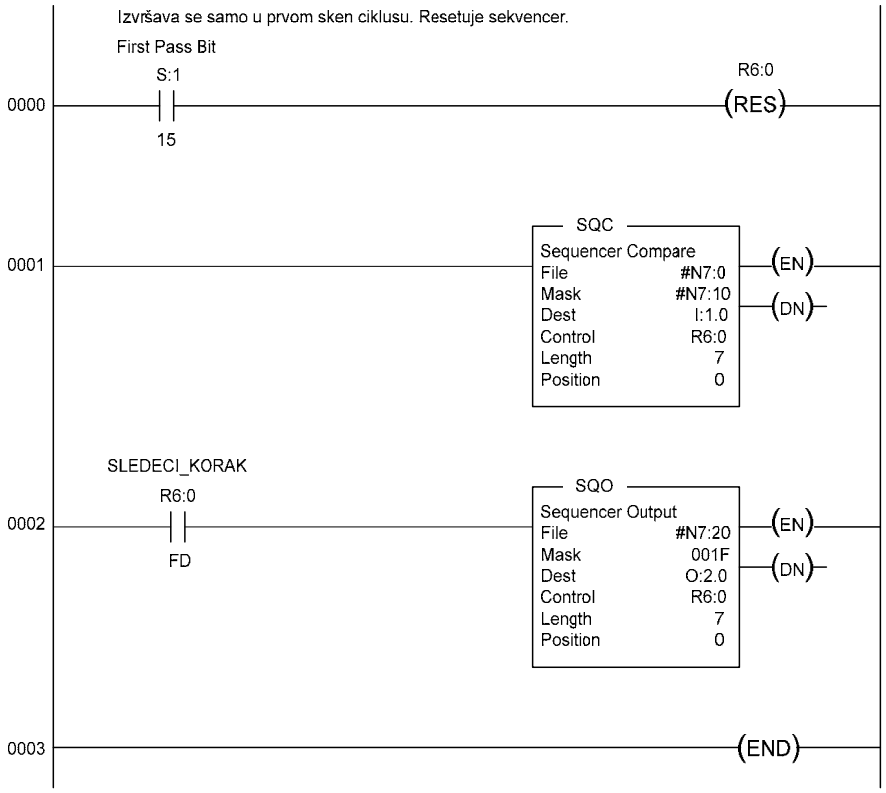
Adresa	15..7	6 START	5 SWC+	4 SWC-	3 SWB+	2 SWB-	1 SWA+	0 SWA-
N7:1	0..0	1	0	0	0	0	0	0
N7:2	0..0	0	0	0	0	0	1	0
N7:3	0..0	0	0	0	1	0	0	0
N7:4	0..0	0	1	0	0	0	0	0
N7:5	0..0	0	0	0	0	1	0	0
N7:6	0..0	0	0	1	0	0	0	0
N7:7	0..0	0	0	0	0	0	0	1

T. 5 Sadržaj datoteke maski naredbe SQC

Adresa	15..7	6 START	5 SWC+	4 SWC-	3 SWB+	2 SWB-	1 SWA+	0 SWA-
N7:11	0..0	1	0	0	0	0	0	0
N7:12	0..0	0	0	0	0	0	1	0
N7:13	0..0	0	0	0	1	0	0	0
N7:14	0..0	0	1	0	0	0	0	0
N7:15	0..0	0	0	0	0	1	0	0
N7:16	0..0	0	0	1	0	0	0	0
N7:17	0..0	0	0	0	0	0	0	1

T. 6 Sadržaj izvorne datoteke naredbe SQO.

Adresa	15..5	4 MOVE_C-	5 MOVE_C+	4 MOVE_B-	3 MOVE_B+	2 MOVE_A+
N7:21	0..0	0	0	0	0	0
N7:22	0..0	0	0	0	0	1
N7:23	0..0	0	0	0	1	1
N7:24	0..0	0	1	0	1	1
N7:25	0..0	0	1	1	0	1
N7:26	0..0	1	0	1	0	1
N7:27	0..0	1	0	1	0	0



Sl. 4-26 Leder dijagram sekvencera koji koristi naredbe SQC i SQO.