TEHNIČKO VELEUČILIŠTE U ZAGREBU
ELEKTROTEHNIČKI ODJEL

PROGRAMIRLJIVI LOGIČKI KONTROLERI

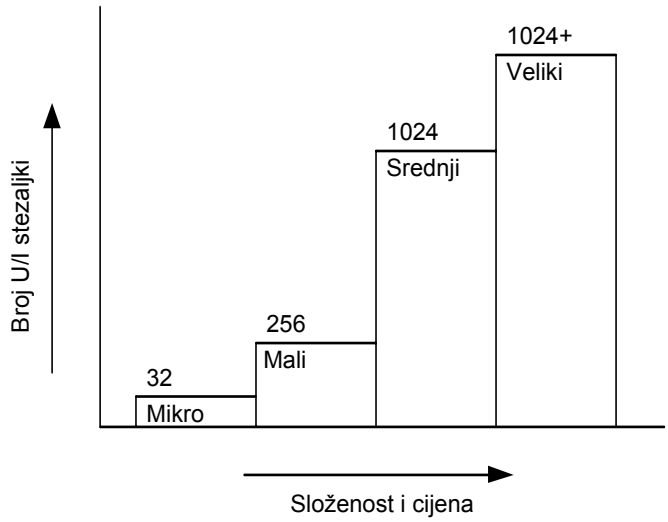
Goran Malčić dip.ing.

SADRŽAJ

1 Uvod

Krajem 60-tih godina industrijski proizvodni pogoni su uglavnom bili upravljani sustavima zasnovanim na relejnim krugovima (relejna logika). Svaki put kada bi se promjenio proizvodni program morale su se napraviti prilagodbe upravljačkih sklopova što bi iziskivalo veliko vrijeme praznog hoda proizvodnje, a s time i velike troškove. Iz toga se vidi da su relejni upravljačjki sustavi bili vrlo nefleksibilni, te da bi se promjenila funkcija upravljanja jednog takvog relejnog sklopa nije svaki put bilo dovoljno promjeniti njegovo ožičenje, nego je ponekad trebalo krenuti sa sastavljanjem novog sklopa. Otprilike u isto vrijeme je i razvoj mikroprocesora došao do određenog nivoa te se pojavila ideja o izradi elektroničko-kompjuterskog upravljačkog sustava koji bi se jednostavno sa promjenom proizvodnog programa dao reprogramirati. Tada su napravljeni prvi programabilni logički kontroleri skraćeno PLC i vrlo brzo pokazali izuzetne prednosti u odnosu na relejni upravljački sustav – pouzdaniji je od relejnog sustava jer nema mehaničkih pokretnih dijelova; fleksibilniji jer ga pri promjeni proizvodnje treba samo reprogramirati, a ne mijenjati ožičenje; smanjeni je opseg ožičenja i greške u ožičenju; dimenzije su višestruko manje jer su vremenski releji, brojači i ostale relejne upravljačke komponente riješene softverski. Osim toga PLC kao industrijsko računalo otporan je na razne nepovoljne utjecaje iz proizvodnje kao što su prašina, vlaga, visoka temperatura, vibracije, elektromagnetski utjecaji jer je samim svojim ustrojem napravljen tako da se postavi u neposrednoj blizini procesa kojim upravlja. Kao takvi sa visokim stupnjem fleksibilnosti PLC-ovi su vrlo brzo bili široko prihvaćeni.

Danas se programiranje PLC-a se najčešće provodi preko PC IBM računala u jednom od tri PLC programska jezika: Ladder dijagrami, Statement liste te programiranje pomoću funkcijskih blokova. Sve tri metode programiranja imaju svoje prednosti i nedostatke te su podjednako prihvaćene među inženjerima. Sa namjerom da se te metode i općenito rad PLC-a malo pobliže objasni napisana je ova skripta. Skripta je prvenstveno namjenjena studentima koji slušaju kolegij Procesna računala na Elektrotehničkom odjelu Tehničkog veleučilišta, ali može poslužiti kao priručnik i ostalim tehničarima koji se žele baviti ovom tematikom.



PLC uređaje grubo možemo podijeliti prema broju ulazno/izlaznih stezaljki tako da dobijemo četiri kategorije (slika).

S povećanjem ulazno/izlaznih stezaljki mora se povećati i snaga procesora, količina memorije kao i sama složenost uređaja i cijena pa se može reći da su veći PLC uređaji i bolji, ali nije uvijet. U podijeli treba uzeti u obir da li PLC osim digitalnih (diskretnih) ima analogne ulazno/izlazne stezaljke, mogućnost izvođenja matematičkih operacija nad realnim brojevima (float point), PID regulaciju, mogućnost proširenja, itd. Naime spajanjem više manjih PLC uređaja u mrežu može se postići efekt kao da imamo veliki uređaj što je skuplje riješenje, ali nekad neizbježno (velike udaljenosti u proizvodnoj liniji).

Projektant upravljačkih sustava uvijek postavlja pitanje kada u manjim sustavima upotrijebiti PLC umjesto relejnog sklopa. Nekad je to pitanje bilo opravdano visokom cijenom PLC-a, ali danas s obzirom na masovnu proizvodnju i nisku cijenu uporaba mikro PLC-ova je svakako opravdana. Dovoljno je da u sustav upravljanja treba ugraditi nekoliko upravljačkih i vremenskih releja pa da se sa PLC-om dobije jeftiniji i fleksibilniji upravljački sustav svakako podesniji od relejnog upravljačkog sklopa. Navesti ću samo neke od najznačajnijih prednosti PLC-a pred relejnim upravljanjem:

Pouzdanost – nema mehaničkih pokretnih dijelova, izuzetno otporan na razne mehaničke, elektromagnetske utjecaje, te općenito otporan na pogonske uvjete rada, ako nestane napajanje ništa se ne mijenja i kada se napajanje vrati PLC nastavlja sa radom. Greške u ožičenju svode se na minimum s obzirom da se ožičenje PLC-a svodi na ožičenje njegovih ulaza i izlaza;

Adaptivnost - kad se napiše i testira, PLC program za upravljanje nekog uređaja može se bez problema prenjeti na drugi PLC u drugom uređaju. U slučaju identičnih uređaja ili uređaja gdje se zahtijevaju manje izmjene programa to dovodi do smanjenja vremena programiranja i vremena za otklanjanje grešaka.;

Fleksibilnost – za izmjenu programa potrebno je vrlo malo vremena. Izvođači upravljačkog sustava mogu bez problema poslati korisniku izmjenu programa na bilo kojem mediju (disketa npr.) ili putem modema direktno u PLC bez da šalju tehničara za održavanje na lokaciju korisnika.Korisnik može jednostavno prenjeti program u PLC i izvršiti eventualno manje promjene;

Naprednija funkcionalnost – PLC programske aplikacije mogu se sastojati od jednostavnih akcija ponavljanja neke automatske radnje do kompleksne obrade podataka i složenih upravljačkih sustava. Upotreba PLC-a u upravljačkim sustavima nudi projektantima takvih sustava i osoblju u održavanju brojne mogućnosti neizvedive pomoću standardnog relejnog upravljanja;

Komunikacija – sa operatorskim panelima, drugim PLC uređajima i nadzornim upravljačkim računalima olakšava prikupljanje podataka s uređaja i obradu prikupljenih informacija;

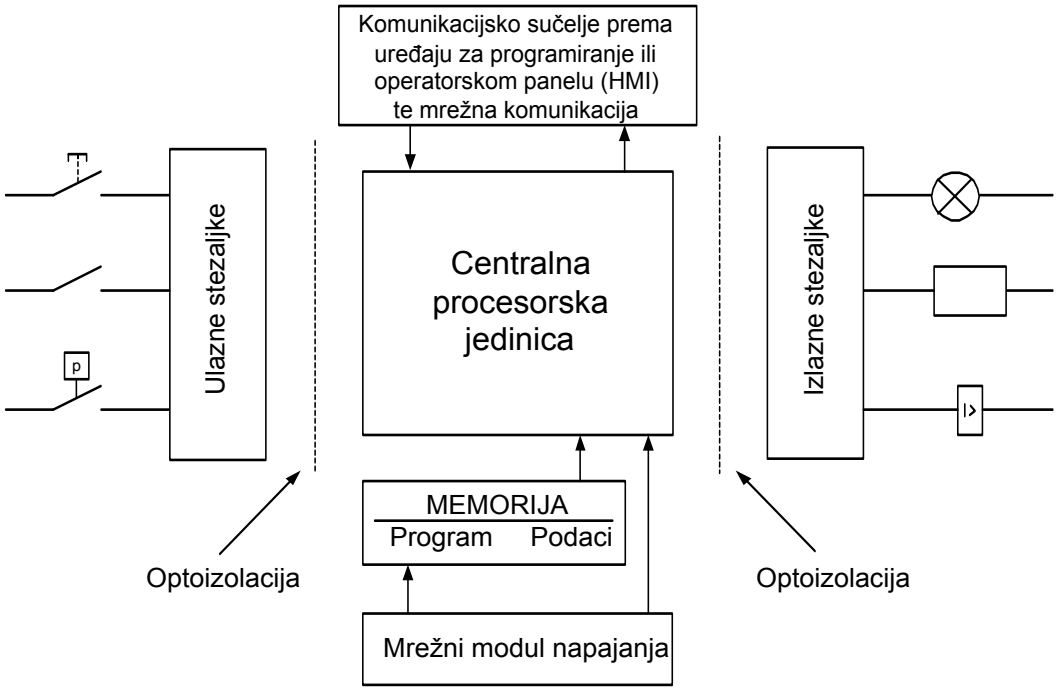
Brzina – brojne aplikacije na automatiziranim strojevima zahtjevaju vrlo brzu reakciju na pojavu nekog signala. Takve aplikacije jednostavno su izvedive uz pomoć PLC-a, a vrlo teško i složeno putem relejnog upravljanja;

Dijagnostika – pomoću funkcija za otklanjanje pogrešaka i dijagnostiku, PLC-i nude brzo i jednostavno otklanjanje softverskih i hardverskih (sklopovskih) grešaka upravljačkog sustava;

2 Princip rada PLC-a

Da bi se objasnio način rada PLC uređaja potreban je kratak pregled njegovih osnovnih cjelina. Svi PLC uređaji od mikro PLC-a do najvećih PLC sustava od preko 1000 U/I signalaimaju, u principu, istu hardversku strukturu, odnosno iste osnovne cjeline:

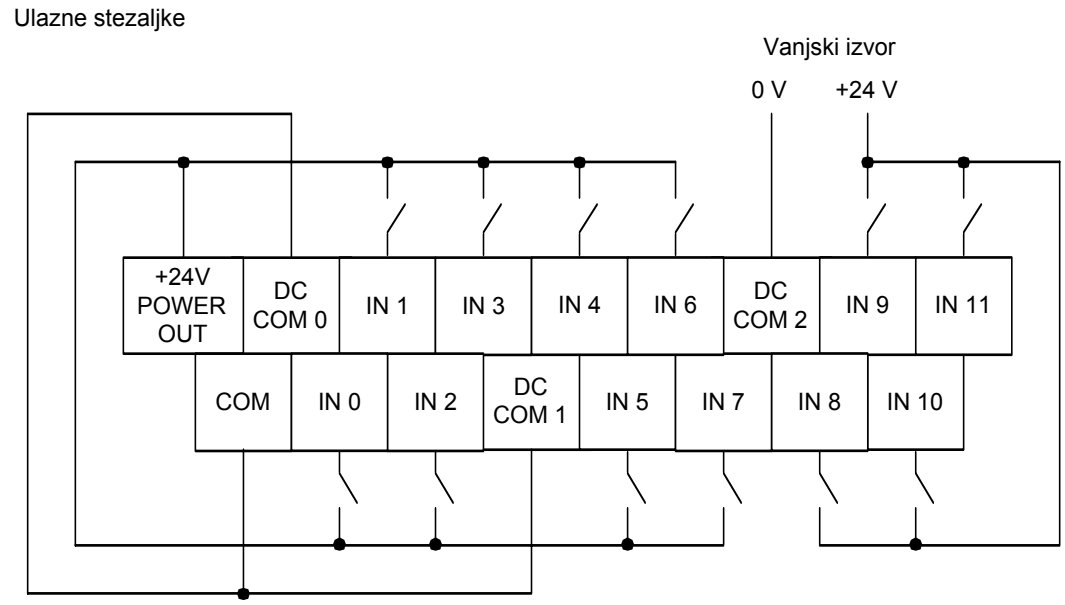
- ulazni dio (digitalni, analogni ulazi)
- izlazni dio (digitalni, analogni izlazi)đž
- CPU, tj. Centralnu procesorsku jedinicu
- memorijski blok za program i podatke
- mrežni dio za napajanje te komunikacijsko sučelje
- moduli za proširenje



Slika – Osnovne cjeline PLC uređaja

2.1 Ulazni dio

Ulazni dio PLC-a su priključne vijčane stezaljke na koje se spajaju dojavni signali iz procesa čijim se radom upravlja, te su mjesto od kojeg počinje prilagodba vanjskog signala iz radne okoline, signalu kojeg razumije procesorska jedinica PLC-a. Informacije koje PLC prima na svojim ulaznim stezaljkama mogu biti digitalne (diskretne) i analogne. Digitalna ulazna informacija može biti npr. signal s krajnje sklopke, senzora, tipkala i sl. dok analogna ulazna informacija može biti npr. naponski signal 0-10 VDC s mjernog pretvornika tlaka,temperature i sl. Za digitalnu informaciju visoko stanje iznosi 14-30 VDC, a nisko stanje 0-5 VDC. Analogna informacija može biti u raznim oblicima – strujni 0-20 mA, strujni 4-20 mA, naponski 0-10 VDC, naponski –10 - +10 VDC uz određenu rezoluciju (8 ili 16 bitni A/D pretvornik). Prilagodba signala s uobičajenog ulaznog napona od 120-230 VAC ili 24 VDC na 5 VDC, tj. naponski nivo logike procesorske jedinice, uključuje optoizolaciju signala, što je vrlo važno kako bi se galvanski odvojili strujni krugovi, čime se spriječava protok struje uslijed potencijalnih razlika strujnih krugova interne logike PLC-a i ulaznog kruga, te filtriranje signala kako bi se smanjile visokofrekventne smetnje, odnosno smetnje uslijed statičkih pražnjenja.

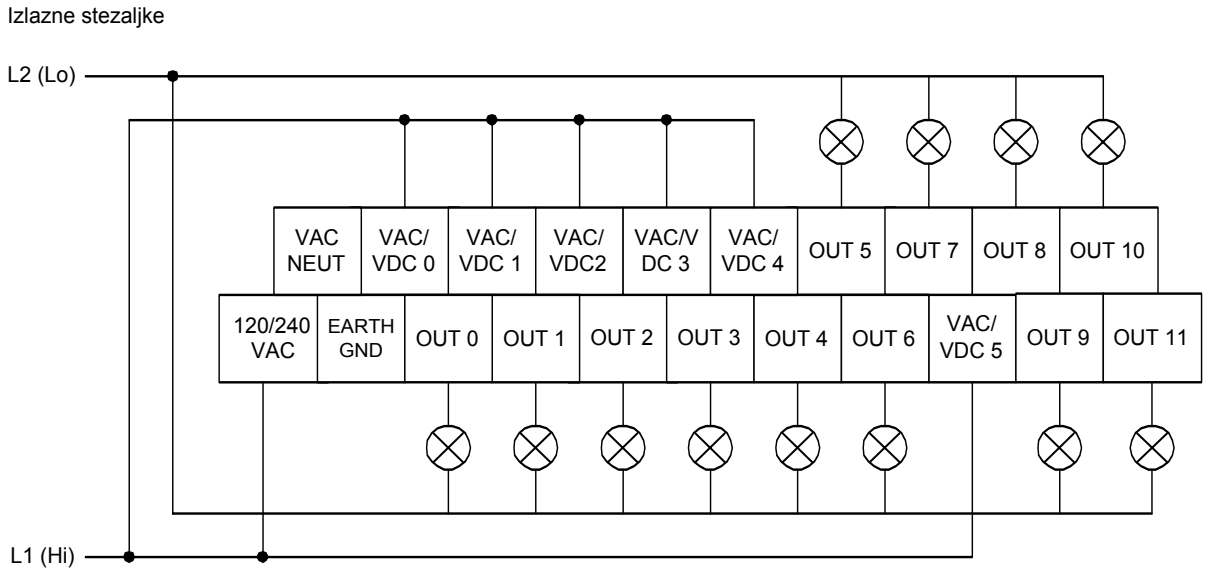


Na slici je primjer ožičenja ulaznih stezaljki PLC-a sa 12 digitalnih ulaza koji su podijeljene u 3 grupe prema stezaljci mase (COM 0 – input 0 do 3, COM 1 – input 4 do 7, COM 2 – input 8 do 11). Stezaljke od IN 0 do IN 7 koriste izvor napajanja iz samog PLC-a dok stezaljke od IN 8 do IN 11 koriste vanjski izvor napajanja. Vanjski izvor napajanja se koristi ako izvor u PLC-u ne može zadovoljiti potrebe senzorske opreme za količinom el. energije. Kod odabira izvora treba uzeti u obzir i pad napona u vodičima ako su udaljenosti između senzora i PLC-a velike.

2.2 Izlazni dio

Izlazni dio PLC-a su priključne vijčane stezaljke na koje se spajaju izvršni uređaji iz procesa kojima PLC šalje digitalne i analogne signale te na taj način upravlja procesom. Na digitalne izlaze iz PLC-a su najčešće spojeni magnetni svici, releji, sklopnici, motorske sklopke, signalne lampe, pneumatski razvodnici i sl., dok na analognei izlaze mogu biti spojeni npr. strujni signal za prikaz neke veličine na pokaznom instrumentu, referenca brzine za frekvencijski pretvarač, PID regulirana veličina itd. Izlazne stezaljke također su optoizolirane od procesorske jedinice radi galvanske izolacije električnih krugova. Digitalni izlazi najčešće su izvedeni kao relejni, tranzistorski ili pomoću trijaka, a svaki od njih ima svoje prednosti i mane:

- relejni izlazi mogu se koristiti za sklapanje istosmjernih i izmjeničnih tereta, za struje do nekoliko ampera. Releji dobro podnose naponske udare i obzirom na zračni razmak između njihovih kontakata ne postoji mogućnost pojave pulzirajućih struja. Releji su međutim relativno spori prilikom sklapanja te imaju vijek trajanja (mjeren maksimalnim brojem sklapanja) manji od trijaka i tranzistora.
- tranzistorski izlazi služe za sklapanje istosmjernih tereta, nemaju pokretnih dijelovakoji se troše i bešumni su. Vrijeme reakcije im je brzo, ali mogu sklapati uglavnom struje do 0.5 A.
- izlazi sa trijacima služe za sklapanje izmjeničnih tereta, a karakteristike su im slične kao tranzistorima.



Na slici je primjer ožičenja izlaznih stezaljki PLC-a sa 12 digitalnih izlaza koji su podijeljene u 6 grupa prema stezaljci napajanja (VAC/VDC 0 – output 0, VAC/VDC 1 – output 1, VAC/VDC 2 – output 2, VAC/VDC 3 – output 3, VAC/VDC 4 – output 4 do 7, VAC/VDC 5 – output 8 do 11). Releji koji se nalaze u PLC-u otvaraju/zatvaraju pojedine strujne krugove te na taj način uključuju/isključuju trošila. Snaga odnosno struja uz određenu voltažu koju releji mogu izdržati navedena je uvijek u specifikacijama PLC-a.

2.3 Centralna procesorska jedinica

Centralna procesorska jedinica s memorijom glavna je jedinica PLC uređaja. Najkraće rečeno procesorska jedinica čita stanja svih ulaza PLC uređaja (analognih i digitalnih), logički ih obrađuje u skladu s programom izrađenim od strane korisnika, te upravlja izlazima prema rezultatima dobivenim nakon logičke obrade.

2.4 Memorijski blok za program i podatke

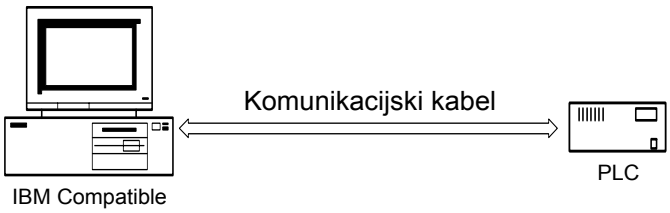
PLC korisnik prilikom programiranja koristi dva segmenta memorije procesorske jedinice – programske datoteke i datoteke podataka. Programske datoteke koriste korisnički definirane programe, potprograme i datoteku za dojavu i obradu grešaka. Datoteke podataka služe za memoriranje programski ovisnih podataka kao što su U/I status, postavne i trenutne vrijednosti brojača i vremenskih članova te ostale memorijske konstante i varijable. Podaci programske datoteke i datoteke podataka pohranjuju se u dvije vrste memorije; RAM (eng. random access memory – memorija s izravnim pristupom) i EEPROM (eng. electricaly erasable programable read only memory – električki obrisiva programabilna memorija namjenjena isključivo za čitanje). RAM memorija u PLC uređajima obično je podržana baterijom kako se po nestanku napona napajanja ne bi izgubili podaci (koji se ipak mogu izgubiti ako se istroši baterija), dok EEPROM memorija trajno sprema podatke bez obzira na napon napajanja. Korisnički programi izvode se iz RAM memorije, a dobra je preksa da se pohrane i u EEPROM memoriji te da se učitavaju u RAM svaki put kada se uključuje PLC, ili u slučaju gubitka podataka iz RAM memorije (iz bilo kojeg razloga). Sistemski program i memorija za upravljanje radom PLC uređaja nisu vidljivi i dostupni korisniku, ali su od ključne važnosti za njegov učinkovit rad.

2.5 Modul napajanja

Kao i na svakom računalu modul napajanja je najrobusniji i najteži njegov dio. Neosjetljiv je na smetnje koje dolaze iz električne mreže kao i na kraće ispade mrežnog napona (trajanja 10-15 ms). Standardni ulazi napajanja PLC uređaja su: 120/230 VAC i 24 VDC.

2.6 Komunikacijsko sučelje

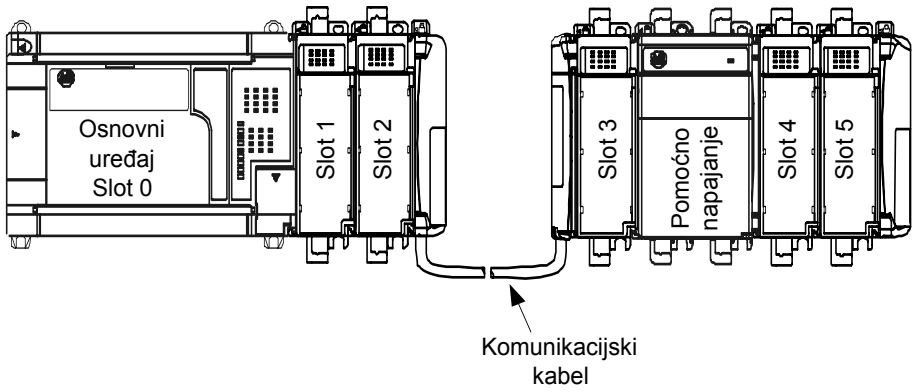
Komunikacijsko sučelje ima višestruku namjenu. Prva i osnovna je komunikacija sa nadređenim PC računalom na kojem se piše upravljački program, šalje u PLC te dijagnosticira stanje rada (slika).



Ostale mogućnosti su komunikacija sa ostalim PLC uređajima i raznim senzorima preko njihove interne mreže (npr DeviceNet), komunikacija sa raznim vrstama operatorskih panela te komunikacija modemomskom vezom. Gotovo svi PLC uređaji imaju ugrađen serijski port za komunikaciju (RS-232 – električki standardi), a komunikacija se vrši preko protokola koji ovisi o proizvođaču uređaja (najčešće full duplex serijska veza).

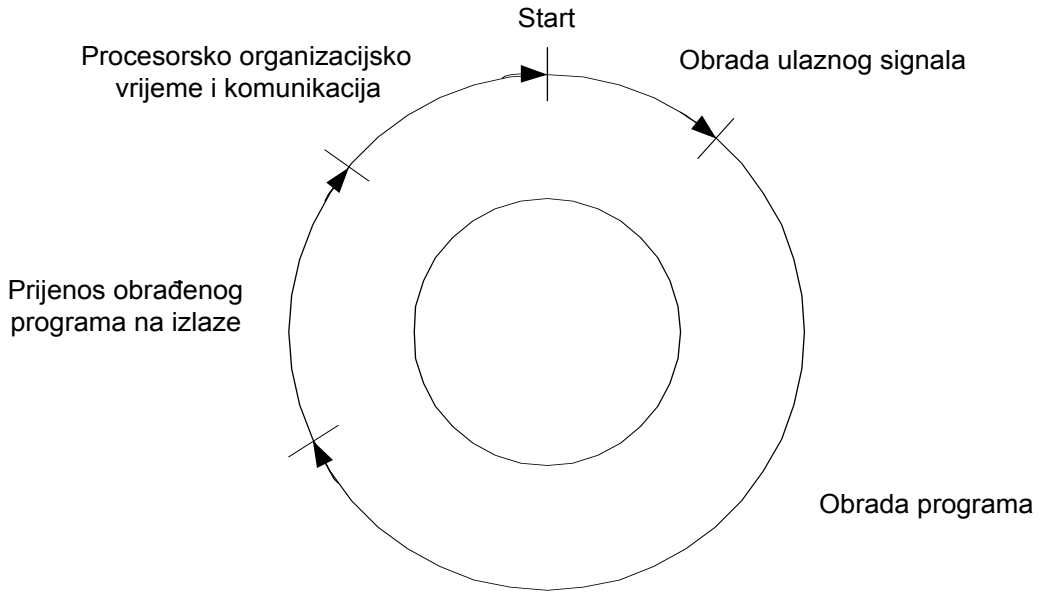
2.7 Moduli za proširenje

U osnovi PLC uređaj je od jednog dijela te na sebi ima ograničeni broj ulaznih i izlaznih stezaljki. Kada je za proces potrebno više ulaza ili izlaza nego ih na sebi ima osnovni uređaj koriste se moduli za proširenje (slot). Modul za proširenje je poseban uređaj koji se spaja na PLC i koji na sebi ima dodatne ulazne i/ili izlazne stezaljke. Na taj način se PLC uređaj uvijek može proširiti bez da se nabavlja novi. Najčešće se moduli za proširenje prodaju kao moduli za digitalne ulaze i/ili izlaze te moduli za analogne ulaze i/ili izlaze.



Moduli se napajaju el.energijom iz osnovnog uređaja, ali mogu koristiti i posebna napajanja (slika). Preporuča se da osnovni uređaj i moduli za proširenje koriste isti izvor napajanja. U pogonu moduli mogu biti udaljeni od osnovnog uređaja te se veza ostvaruje komunikacijskim kabelom. Broj modula koji se mogu spojiti osnovni uređaj ovisi o proizvođaču.

2.8 Rad uređaja



Slika – Ciklus rada PLC-a

Sam rad uređaja najzornije je prikazan slikom. Pošto PLC prema promjeni stanja na njegovim ulazima mora kontinuirano korigirati stanja izlaza kako je to određeno logikom u korisničkom programu on tu internu obradu podataka vrti ciklički u beskonačnoj petlji. U osnovi ciklus obrade podataka podijeljen je na nekoliko dijelova:

1. Obrada ulaznog stanja – očitanje stanja ulaza te prijenos podataka ulaznog stanja u ulazni memorijski registar procesorske jedinice;

2. Obrada programa – programska obrada ulaznih stanja prema logici korisničkog programa te slanje rezultata u izlazni memorijski registar procesorske jedinice;

3. Prijenos obrađenog programa na izlaze – prijenos obrađenih podataka iz izlaznog memorijskog registra na fizičke izlaze PLC-a;

4. Procesorsko organizacijsko vrijeme i komunikacija – odvijaju se operacije potrebne za funkcioniranje operativnog sustava PLC uređaja te komunikacija sa vanjskim jedinicama.

Vrijeme jednog ciklusa za oko 500 programskih naredbi se kreće oko 1,5 ms.

2.9 Adresiranje

Svi podaci u memoriji PLC-a su svrstani u tri osnovne grupe zapisnika koji se dijele na podgrupe. Navešćemo one najčešće, a radi lakšeg razumjevanja imena zapisnika (file) su na engleskom jeziku.

DATA FILES

(podatkoivni zapisnici):

- Output file – sprema vrijednosti koje su očitane za vrijeme pretraživanja stanja izlaznih stezaljki PLC-a (stanje izlaza)
- Input file - sprema vrijednosti koje su očitane za vrijeme pretraživanja stanja ulaznih stezaljki PLC-a (stanje ulaza)
- Bit file – sprema vrijednosti veličine bita (markere, flag)
- Timer file – prati i sprema vrijednosti timera iz kontaktnog dijagrama
- Counter file – sprema vrijednosti brojača iz kontaktnog dijagrama
- Integer file – sprema podatak maksimalne veličine 16 bita
- Long word file - sprema podatak maksimalne veličine 32 bita
- Float file – može spremiti realni broj veličine 32 bita
- Status file – sprema podatke o radu sistema i greškama u radu (bitno jer se prema broju greške vrlo lako obavlja dijagnostika sustava)
- Message file – sprema podatke o stanju poruka između PLC uređaja

FUNCTION FILES

(funkcijski zapisnici):

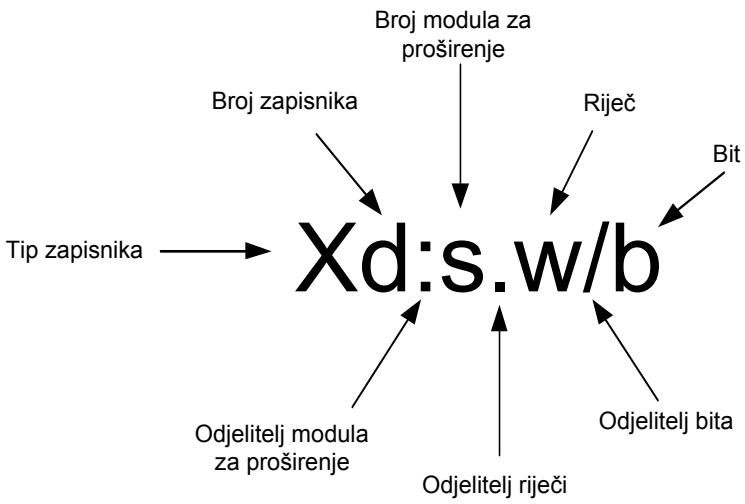
- Base hardware information file – sadrži tehnički opis PLC-a
- I/O status file – sadrži informacije o modulima za proširenje
- Communications status file – sadrži informacije o komunikacijskim parametrima i komunikacijskoj aktivnosti
- Real time clock – vrijeme
- Trim pot information – sadrži informacije o vrijednosti trimer potenciometra ako ih PLC posjeduje (većina da)
- Memory module – sadrži informacije o naknadno ugrađenoj memorijskoj kartici (nemaju svi modeli PLC-a, uglavnom EEPROM)

PROGRAM FILES

(programski zapisnici):

- Program files – programski kod napisan u nekoj od tehnika programiranja npr. kontaktni dijagrami
- System file – sistemski zapisnici
-

Memorijski registri kojima se barata pri programiranju PLC-a su veličine bita i veličine riječi (16 i 32 bitne riječi). Da bi mogli programirati mora se znati način na koji se u kontaktnom dijagramu ili nekom drugom PLC programskom jeziku označavaju ulazi, izlazi i ostale memorijske lokacije. U osnovi svi proizvođači PLC-a na više ili manje modificirani način se služe sustavom adresiranja prema slici.



Primjeri:

Adresa	Tip zapisnika	Broj zapisnika	Broj modula	Riječ	Bit
I:0/3	Input	-	0 (osnovni)	0	3
O:4/6	Output	-	4	0	6
O:2.1	Output	-	2	1	-
I:3.2	Input	-	3	2	-
I:1.2/12	Input	-	1	2	12
B3:5/9	Bit	3	-	5	9
N7:3	Integer	7	-	3	-

3 Programiranje

Jednako kao i ostala industrijska računala i PLC izvodi program i prema njemu upravlja procesom, odnosno kontrolira ulaze i upravlja izlazima. Pisanje programa najćešće se izvodi preko nadređenog PC računala na kojem je instaliran softver za korišteni PLC-a. Svaki proizvođač uz svoj PLC daje softver koji je u stvari kombinacija programskog editora, compilera te komunikacijskog softvera. U editoru se napiše programski kod u nekom od programskih jezika te se zatim provjeri njegova sintaksa (sve u PC računalu). Ako program nema sintaksnih grašaka softver ga šalje u RAM memoriju PLC-a te je on spreman za rad. Komunikacija između PC računala i PLC-a je najćešće serijska (RS-232) te može biti aktivna i za vrijeme izvođenja programa na PLC-u. Na taj način na ekranu PC računala uvijek možemo pratiti stanja ulaza i izlaza te zadavati naredbe direktno preko tipkovnice i miša.

PLC se također može programirati i preko ručnih programatora koji posjeduju mali LCD displej uz skromnu tipkovnicu. Takvi se uređaji direktno spajaju na PLC te se mogu koristiti za kraće programe ili za manje izmjene programa kada se to mora obaviti u pogonu. Za neke jednostavnije procese postoje PLC uređaji koji posjeduju na sebi i display i par funkcijskih tipki pa se mogu programirati na licu mjesta.

Kako bi uspješno proveli programiranje PLC-a koji će potom upravljati procesom moramo na neki način program ispitati. Ispitivanje programa možemo vršiti samo tako da na ulaze u PLC dovedemo stanje veličina iz realnih uvjeta u procesu. Za to se koriste tzv. simulatori stanja PLC-a. Simulator stanja nije ništa drugo nego niz prekidača i kontrolnih lampica koji se zasebno spoje na ulaze i izlaze PLC-a. Kada želimo simulirati rad nekog senzora bitnog za proces npr. protusmrzavajući senzor (preklopi kada je temperatura < 0°C) na mjestu njegovog ulaza u PLC preklopimo prekidač koji će umjesto njega simulirati da je temperatura ispod 0°C. Jednako tako će kontrolna lampica svijetliti kada je aktiviran izlaz na koji je ona spojena. Na taj način dobivamo simulaciju rada PLC-a vjernu stvarnim uvjetima u procesu kako bismo mogli ispitati program. Programiranje i ispitaivanje se vrši za radnim stolom, kada je testiranje gotovo PLC se odnese u pogon i ugradi na stroj. Također neki proizvođači nude mogućnost simulranja stanja softverski što je isto dobro kada su u pitanju manji procesi (nema potrebe za izradom simulatora).

Neki PLC uređaji su opremljeni izmjenjivim EEPROM memorijskim karticama. To mnogo olakšava stvar pri npr. promjeni proizvodnog programa u nekom pogonu. Dovoljno je na trenutak izgasiti PLC, izvaditi staru EEPROM memorijsku karticu i umetnuti novu sa drugim programom. Nakon uključenja PLC radi po programu napisanom na novoj kartici. Stare kartice se mogu reprogramirati i kasnije upotrijebiti pri sljedećoj promjeni.

Proizvođači PLC-a nude razne programske jezike. Najčešće upotrebljavan PLC programski jezik su kontaktni dijagrami (eng. ladder diagram, njem kontaktplan). Ovaj način programiranja prilagodba je relejnih upravljačkih shema (i njihovog grafičkog izgleda) principima rada PLC uređaja. Iz povjesnih razloga je ovaj način programiranja korišten kod prvih PLC-ova, kako bi se osoblje naviklo na izradu relejnih upravljačkih shema najlakše obučilo izradi programa. Kasnije se pokazalo da je ovaj 'grafički' način programiranja lako shvatljiv i onima koji se nisu bavili relejnim upravljanjem pa je to vjerovatno najveći razlog zbog čega je ostao i dan danas nepromjenjen i najšire

prihvaćen među tehničarima. Od PLC programskih jezika uz kontaktne dijagrame tu se nalaze i STL (statement list) instrukcojske liste - programiranje na nivou asemblera i funkcijsko blokovski dijagrami (grafičko programiranje). Sva tri spomenuta PLC programska jezika su definirana IEC 1131-3 međunarodnim normama koje se bave načinom programiranja PLC uređaja. Neki proizvođači mikro PLC-a nude mogućnost programiranja i pomoću BASIC i C programskih jezika no ti programski jezici nemaju široku zastupljenost u ovom djelu primjene računala.

4 STL

4.1 Opis

STL (statement liste) je programski jezik koji omogućava programerima da služeći se jednostavnim naredbama na nivou asemblera opisuju operacije koje treba izvršiti PLC da bi upravljao procesom (kao i kod programiranja u asembleru svaki redak programskog kod-a predstavlja jednu naredbu za procesor). Modularna građa ovog jezka omogućuje nam da riješavamo kompleksne probleme na najučinkovitiji način. STL programski jezik ima određenu hijerarhiju pa nam ga je prema njoj najlakše opisati:

PROGRAM

KORAK (STEP)

NAREDBA

UVJETNI DIO

IZVRŠNI DIO

Svaki STL program se sastoji od niza koraka koji se izvršavaju slijedom od prvog prema zadnjem. Riječ korak se može tumačiti na više načina, ali većina STL programa korak tumači kao zasebnu naredbu. Korak ustvari predstavlja logički blok unutar kojega je napisan neki programski kod i kada se izvršavaju naredbe iz tog logičkog bloka kažemo da se izvršava određeni korak. U STL programskom jeziku se može programirati i bez koraka odnosno da cijeli program bude jedan korak, ali je to prilično nezgodno jer se teže kontrolira tok informacije,a i ne može se u potpunosti iskoristiti mogućnost programskog jezika. Svaki korak ima svoju oznaku (label) po kojoj ga se unutar programa može pozvati.

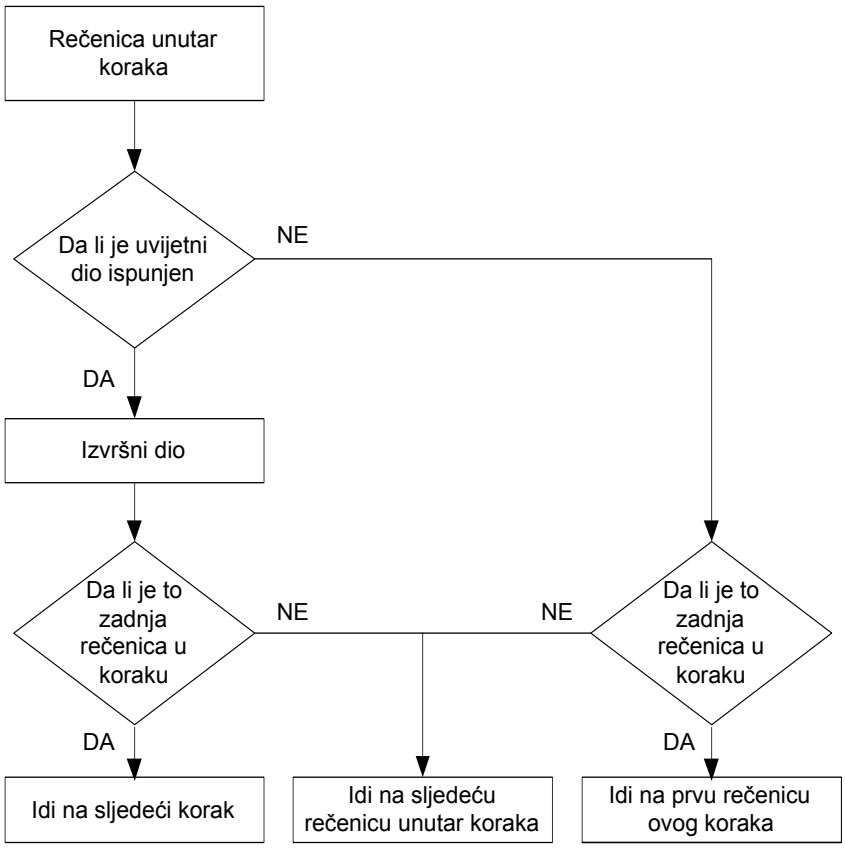
Naredbe su najosnovniji dio programa i zadaju se tako da imaju uvijetni i izvršni dio pa se u informatičkom jeziku takav raspored naziva rečenica (sentence).

Uvijetni (conditional) dio je lista jednog ili više uvijeta čije se stanje provjerava u toku programa (stanje ulaza i izlaza,brojača, timera, itd). Uvjetni dio uvijek počinje riječju IF iza koje slijede uvijeti koji na kraju daju rezultat – visoko ili nisko. Ako su svi uvijeti ispunjeni uvijetni dio šalje logičko stanje visoko u izvršni (executive) dio koji potom izvršava zadane zadatke (npr. upravlja izlaznim stezaljkama PLC-a). Izvršni dio uvijek počinje riječju THEN. Na slici je prikazan STL programski kod koji se može nalaziti unutar jednog koraka.

IF			I:0/1	ako je ulaz 0/1 aktivan (visoko)
THEN	SET		O:0/2	onda uključi izlaz 0/2
IF		N	I:0/4	ako je ulaz 0/4 nije aktivan
THEN	SET		O:0/7	onda uključi izlaz 0/7
IF		N	I:0/2	ako je ulaz 0/2 nije aktivan
	OR		O:0/4	ili je izlaz 0/4 aktivan
THEN	SET		O:0/3	onda uključi izlaz 0/3
	RESET		O:0/5	onda isključi izlaz 0/5

Prva i druga rečenica su jednostavne i sastoje se samo od jednog uvijeta i jedne izvršne naredbe. U drugoj rečenici imamo dva uvijeta određena ILI (OR) logičkom funkcijom koji ako su ispunjeni upravljaju radom izlaznih stezaljki O:0/3 i O:0/5.

Na ovaj način možemo napraviti cijeli upravljački program odnosno da su sve rečenice dio jednog koraka, no da bi programiranju pristupili imalo ozbiljnije moramo se koristiti koracima. Programiranje sa koracima ima jedno pravilo sa kojim možemo određivati tok izvođenja programa. Pravilo govori da ako uvijetni dio zadnje rečenice unutar koraka nije ispunjen odnosno ako se izvršni dio zadnje rečenice ne izvršava program ne ide na sljedeći korak nego se vraća na prvu rečenicu unutar istog koraka. To bi značilo iz primjera sa slike da ako bi I:0/2 bio aktivan i O:0/4 bio neaktivan, OR funkcija bi davala logičko stanje nisko i izvršni dio ne bi radio, a program ne bi prešao na sljedeći korak nego bi se vratio na prvu rečenicu istoga odnosno provjeravao stanje I:0/1 itd. Pravilo ćemo najlakše prikazati na blokovskom dijagramu.



Dakle program će se vrtiti u petlji unutar jednog koraka sve dok se uvijet iz zadnje rečenice ne zadovolji. No nije uvijek poželjno da se program vrti unutar jednog koraka pa je zato naprevljena funkcija NOP (primjer).

```
STEP 12
IF                                  I:0/8    ako je ulaz 0/8 aktivan (visoko)
THEN        RESET                   O:0/2    onda isključi izlaz 0/2

IF                          N       O:0/1    ako je izlaz 0/1 nije aktivan
THEN        SET                     O:0/9    onda uključi izlaz 0/9

IF          NOP                              uvijek u stanju visoko
THEN        SET                     B3:0/0   onda setiraj bit B3:0/0
            LOAD                    N7:1     učitaj tu vrijednost u radni registar
            TO                      N7:5     i spremi ga na adresu N7:5
```

Iz danog primjera korak br.12 ima u uvijetnom dijelu zadnje rečenice funkciju NOP koja uvijek daje stanje visoko. Zahvaljujući toj funkciji kada izvođenje programa dođe do zadnje rečenice unutar koraka uvijet će uvijek biti zadovoljen te će program skočiti na sljedeći korak. U danom primjeru stanje bita na adresi B3:0/0 uvijek će biti '1'.

Treba primjetiti naredbu LOAD TO koja vrijednost sa adrese N7:1 učitava u pomoćni memorijski registar (radni registar – akumulator) te ju potom iz njega sprema na adresu N7:5. Akumulator je pomoćni memorijski spremnik koji koristimo da smjestimo neku vrijednost u njega u međukoraku naredbe (32 bita).

4.2 Naredbe

Naredba pridruženja

Naredba pridruženja (znak jednakosti =) je izvršna naredba koja postavlja izlaz veličine bita u stanje ovisno o stanju uvijetnog dijela naredbe.

Primjer:

IF		I:0/1	ako je ulaz 0/1 aktivan
	AND	O:0/8	i aktivan izlaz 0/8
	OR	B3:0/3	ili aktivan marker B3:0/3
THEN	=	O:0/4	upravljaj izlazom 0/4

Ako je uvjetni dio naredbe zadovoljen izlaz O:0/4 će skočiti u stanje 1 a ako nije pašće u 0. Treba naglasiti da će se promjena dogoditi svaki put kada program izvrši uvijetni dio naredbe.

Naredba SET

Naredba RESET

Naredba OTHRW

Naredba SET se koristi da postavi (setira) vrijednost veličine bita iz logičkog stanja '0' u logičko stanje '1'. Naredba RESET se koristi da postavi (resetira) vrijednost veličine bita iz logičkog stanja '1' u logičko stanje '0'. Naredbom OTHRW nam je omogućeno da ako uvijetni dio nije zadovoljen izvršimo jedan dio izvršnog dijela naredbe.

Primjer:

IF		I:0/1	ako je ulaz 0/1 aktivan
THEN	RESET	O:0/2	onda isključi izlaz 0/2
	SET	O:0/3	onda uključi izlaz 0/3
OTHRW	SET	O:0/4	inače uključi izlaz 0/4

Ako je ulaz I:0/1 aktivan isključiće se izlaz O:0/2 i uključiti O:0/3. Ako ulaz I:0/1 nije akivan samo će se uključiti izlaz O:0/4. Jednom postavljen izlaz npr. O:0/3 u logičko 1 naredbom SET može doći u stanje 0 samo ako se upotrijebi naredba RESET.

Naredba AND – logičko 'I'

Naredba AND je logička naredba koja vrši funkciju 'I' nad vrijednostima veličine bita ili riječi (16, 32 bita).

Primjeri – bit:

IF		I:0/1	ako je ulaz 0/1 aktivan
	AND	I:0/2	ako je ulaz 0/2 aktivan
THEN	=	O:0/2	onda uključi izlaz 0/2

Samo ako su oba ulaza I:0/1 i I:0/2 aktivna aktivirat će se izlaz O:0/2.

- riječ:

								Operand 1 = 45 decimalno
								AND operand 2 = 236 decimalno
								Rezultat = 44 decimalno

Naredba AND nad podatkom veličine riječi može se upotrijebiti na dva načina:

1. npr. IF (N7:1 AND N7:2) = 34 THEN ....

IF			(N7:1	
	AND		N7:2)	
		=	34	ako je rezultat AND naredbe = 34
THEN	.....			

2. npr IF (N7:1 = 34) AND (N7:2 = 34) THEN ....

IF			(N7:1	
		=	34)	ako je istina
	AND			
			(N7:2	
		=	34)	ako je istina
THEN	.....			

Samo ako su oba uvijeta zadovoljena izvršit će se izvršni dio.

Naredba OR – logičko 'ILI'

Naredba OR je logička naredba koja vrši funkciju 'ILI' nad vrijednostima veličine bita ili riječi (16, 32 bita).

Primjeri – bit:

IF		I:0/1	ako je ulaz 0/1 aktivan
	OR	I:0/2	ako je ulaz 0/2 aktivan
THEN	=	O:0/2	onda uključi izlaz 0/2

Ako je bilo koji od ulaza I:0/1 i I:0/2 aktivan aktivirat će se izlaz O:0/2.

riječ:

								Operand 1 = 45 decimalno
								OR operand 2 = 236 decimalno
								Rezultat = 237 decimalno

Naredba OR nad podatkom veličine riječi može se upotrijebiti na dva načina:

1. npr. IF (N7:1 OR N7:2) = 34 THEN ....

IF			(N7:1	
	OR		N7:2)	
		=	34	ako je rezultat OR naredbe = 34
THEN	.....			

2. npr IF (N7:1 = 34) OR (N7:2 = 34) THEN ....

IF			(N7:1	
		=	34)	ako je istina/laž
	OR			
			(N7:2	
		=	34)	ako je laž/istina
THEN	.....			

Ako je bilo koji od uvijeta zadovoljen izvršiće se izvršni dio.

Naredba XOR – logičko ekskluzivno 'ILI'

Naredba XOR je logička naredba koja vrši funkciju ' Ekskluzivno ILI' nad vrijednostima veličine bita ili riječi (16, 32 bita).

Primjeri – bit:

IF		I:0/1	ako je ulaz 0/1 aktivan
	XOR	I:0/2	ako je ulaz 0/2 aktivan
THEN	=	O:0/2	onda uključi izlaz 0/2

Ako je bilo koji od ulaza I:0/1 i I:0/2 aktivan aktivirat će se izlaz O:0/2 (ne smiju biti oba aktivna).

riječ:

								Operand 1 = 45 decimalno
								OR operand 2 = 236 decimalno
								Rezultat = 193 decimalno

Naredba XOR nad podatkom veličine riječi može se upotrijebiti na dva načina:

3. npr. IF (N7:1 XOR N7:2) = 34 THEN ....

IF			(N7:1	
	XOR		N7:2)	
		=	34	ako je rezultat XOR naredbe = 34
THEN	.....			

4. npr IF (N7:1 = 34) XOR (N7:2 = 34) THEN ....

IF			(N7:1	
		=	34)	ako je istina/laž
	XOR			
			(N7:2	
		=	34)	ako je laž/istina
THEN	.....			

Ako je bilo koji od uvijeta zadovoljen, a nisu oba izvršiće se izvršni dio.

Matematičke naredbe

U STL jeziku možemo koristiti osnovne matematičke operacije (zbrajanje, oduzimanje, množenje, dijeljenje) nad podacima veličine riječi. Stavljamo ih uz naredbu LOAD TO što znači da se uvijek nalaze u izvršnom dijelu naredbe.

Primjer. (N7:0 + 25) * (N7:2-12) i rezultat spremi u N7:3

IF			NOP	
THEN	LOAD		N7:0	učitava N7:0 u radni registar
		+	25	zbraja sadržaj radnog registra sa 25
	TO		N7:1	i sprema u pomoćni marker N7:1
	LOAD		N7:2	učitava N7:2 u radni registar
		-	12	oduzima sadržaj radnog registra sa 12
	TO		N7:3	i sprema u pomoćni marker N7:3
	LOAD		N7:1	učitava N7:1 u radni registar
		*	N7:3	množi sadržaj radnog registra sa N7:3
	TO		N7:3	i sprema u pomoćni marker N7:3

Marker N7:1 smo koristili kao pomoćni marker što je vrlo čest slučaj kod programiranja u STL jeziku. Njegovo stanje nas zanima samo u ovom koraku tako da ga u sljedećim koracima također možemo koristiti kao pomoćni memorijski spremnik. Kod dijeljenja treba spomenuti dvije stvari:

1. ako se dijeli sa nulom PLC će javiti grešku i u principu prestati sa radom
2. PLC dijeli samo cijele brojeve a ostatak sprema u posebni registar (npr. 19 / 5 kao rezultat daće 3, a ostatak 4 će spremiti u za to previđen registar)

Naredbe usporedbe

U STL jeziku imamo osnovne naredbe usporedbe dvaju podataka veličine rijrči, a to su:

> veće	≥ veće ili jednako
< manje	≤ manje ili jednako
= jednako	<> nejednako

Uvijek dolaze u uvijetnom dijelu uz IF THEN naredbu.

Primjer:

IF			(N7:4	ako je vrijednost iz N7:4
		≤	25)	manja ili jednaka od 25
	OR		(N7:5	ili je vrijednost iz N7:5
		<>	6)	nejednaka 6
THEN				

Naredba INV

Naredba INV je izvršna naredba koja od podataku veličine riječi (binarnog broja) radi jedinični komplement.

								Operand = 45 decimalno
								Jedinični komplement

Primjer:

```
IF      NOP
THEN    LOAD    N7:1    učitaj tu vrijednost u radni registar
        INV             napravi njen jedinični komplement
        TO      N7:5    i spremi ga na adresu N7:5
```

U primjeru se od vrijednosti sa adrese N7:1 radi jedinični komplement i sprema se na adresu N7:5. Treba uočiti da je na adresi N7:1 ostala neinvertirana vrijednost.

Naredba CPL

Naredba CPL je izvršna naredba koja podataku veličine riječi mijenja predznak (od binarnog broja radi dvojni komplement)

								Operand = 45 decimalno
								Jedinični komplement
								Zbrojimo jedinicu
								Dvojni komplement = -45 decimalno

Primjer:

```
IF                      ( N7:1
                    <   0 )     ako je vrijednost riječi sa adrese N7:1<0
THEN    LOAD        N7:1        učitaj tu vrijednost u radni registar
        CPL                     napravi njen dvojni komplement
        TO          N7:5        i spremi ga na adresu N7:5
```

U primjeru se nagativnoj vrijednosti sa adrese N7:1 mijenja predznak i sprema se na adresu N7:5. Treba uočiti da je na adresi N7:1 ostala negativna vrijednost.

Naredba BID

Naredba BID konvertira binarni broj veličine riječi u BCD (binary coded decimal) kod.

Primjer:

Dec/Binarno	3547	0000110111011011
BCD	3547	0011 0101 0100 0111

IF	NOP		
THEN	LOAD	N7:1	učitaj tu vrijednost u radni registar
	BID		konvertiraj u BCD kod
	TO	N7:5	i spremi ga na adresu N7:5

Treba naglasiti da ako je veličina N7:1 - 16 bita onda ulazna riječ može imati maksimalnu BCD vrijednost 9999.

Naredba DEB

Naredba DEB konvertira BCD (binary coded decimal) kod u binarni broj.

Primjer:

BCD	2673	0010 0110 0111 0011
Dec/Binarno	2673	0000101001110001

IF	NOP		
THEN	LOAD	N7:1	učitaj BCD vrijednost u radni registar
	DEB		konvertiraj u binarni kod
	TO	N7:5	i spremi ga na adresu N7:5

Treba naglasiti da za 16 bitne riječi (N7:5 u primjeru) maksimalna dozvoljena veličina BCD koda je 9999.

Naredba SHL – lijevi pomični registar (shift registar)

Naredba SHL pomiče (shifts) sadržaj radnog registra (akumulatora) za jedan bit uljevo. Pritom se bit najveće važnosti (most significant bit – bit 15) gubi, a na mjesto bita najmanje važnosti (least significant bit – bit 0) se upisuje 0.

															Akumulator
															SHL naredba
															Akumulator poslije naredbe

Primjer:

```
IF      NOP
THEN    LOAD    N7:4    učitaj vrijednost u radni registar
        SHL             pomakni za jedno mjesto ulijevo
        TO      N7:4    i spremi ga na adresu N7:4
```

Treba primjetiti da smo pomicanjem binarnog sadržaja za jedan bit ulijevo ustvari pomnožili tu vrijednost sa 2. Da smo u primjeru dodali još jednu SHL naredbu vrijednost bi pomnožili sa 4 itd.

Naredba SHR – desni pomični registar (shift registar)

Naredba SHR pomiče (shifts) sadržaj radnog registra (akumulatora) za jedan bit udesno. Pritom se bit najmanje važnosti (least significant bit – bit 0) gubi, a na mjesto bita najveće važnosti (most significant bit – bit 15) se upisuje 0.

															Akumulator
															SHR naredba
															Akumulator poslije naredbe

Primjer:

```
IF      NOP
THEN    LOAD    N7:4    učitaj vrijednost u radni registar
        SHR             pomakni za jedno mjesto udesno
        TO      N7:4    i spremi ga na adresu N7:4
```

Treba primjetiti da smo pomicanjem binarnog sadržaja za jedan bit ulijevo ustvari podijelili tu vrijednost sa 2. Da smo u primjeru dodali još jednu SHL naredbu vrijednost bi podijelili sa 4 itd.

Naredba JMP TO

Naredba JMP TO (jump to – skoči na) se koristi da se promjeni tok izvođenja programa.

Primjer:

```
STEP 12

..... naredbe prije unutar koraka 12

IF                                  I:0/8   ako je ulaz 0/8 aktivan (visoko)
THEN        RESET                   O:0/2   onda isključi izlaz 0/2
            JMP TO                  19      skoči na korak 19

..... naredbe poslije unutar koraka 12



STEP 19

IF                          N       I:0/1   ako je ulaz 0/1 nije aktivan
THEN        SET                     O:0/7   onda uključi izlaz 0/7
            JMP TO                  12
```

Kada tok izvođenja programa dođe na naredbu JMP TO unutar STEP 12 ako je ulaz I:0/8 aktivan programska kontrola prelazi na izvođenje STEP 19 preskačući sve korake između. Ako je u STEP 19 zadovoljen uvijet odnosno I:0/1 nije aktivan kontrola izvođenja programa se vraća na početak STEP 12, a ako nije STEP 19 se vrti dok se uvijet ne ostvari jer je to zadnja (jedina) naredba unutar STEP 19. Dobro je primjetiti da smo naredbom JMP TO preskočili sve naredbe koje dolaze iza nje unutar STEP 12 što ne bi da uvijet I:0/8 nije zadovoljen jer bi se u tom slučaju STEP 12 izvrtio do kraja i prešao na STEP 13.

Naredba INC

Naredba DEC

Naredba INC (increment) povećava vrijednost podatka veličine riječi za 1. Naredba DEC (decrement) smanjuje vrijednost podatka veličine riječi za 1.

Primjer: vrijednost N7:4=35

```
IF                                  I:0/6   ako je ulaz 0/6 aktivan
THEN        INC                     N7:4    povećaj vrijednost za 1 (N7:4=36)
OTHRW       DEC                     N7:4    smanji vrijednost za 1 (N7:4=34)
```

Iz danog primjera se vidi da ako je uvijet ispunjen vrijednost N7:4 će postati 36 , a ako nije ispunjen postaće 34. Naredbe INC i DEC se najćešće koriste kod upotrebe brojača.

Brojači

Brojači su izlazne naredbe koje nam omogućuju kontrolu nad nekim operacijama u procesu koje su vezane za odbrojavanje. Vrijednosti bitne za rad sa brojačima su smještene u counter file, a one se dijele na:

C5:0/EN- Statusni bit brojača je bit koji nam govori da li je određeni brojač aktivan('1') ili nije ('0').

C5:0/DN- Statusni bit brojača je bit koji nam govori da li je određeni brojač završio sa radom

C5:0/PRE - Preset vrijednost veličine riječi je broj koji brojač mora izbrojati da bi završio sa radom

C5:0/ACC – Akumulator - Vrijednost koja uvijek pokazuje koliko je brojač izbrojio impulsa

Primjer:

```
STEP 1
IF          NOP                         uvijek aktivno
THEN        LOAD        10              upiši konstantu 10 u akumulator
            TO          C5:0/PRE        upiši vrijednost iz akumulatora

STEP 2
IF                      I:0/0           ako je ulaz 0/0 aktivan (tipkalo start)
THEN        SET         C5:0            aktiviraj brojač C5:0
            RESET       O:0/0           ugasi izlaz 0/0 (zeleno svijetlo)
            SET         O:0/1           aktiviraj izlaz 0/1 (crveno svijetlo)

STEP 3
IF                      I:0/1           ako je ulaz 0/1aktivan
THEN        INC         C5:0            povećaj vrijednost brojača za 1

STEP 4
IF                      C5:0/DN         ako je C5:0 izbrojao svih 10
THEN        SET         O:0/0           aktiviraj izlaz 0/0 (zeleno svijetlo)
            RESET       O:0/1           ugasi izlaz 0/1 (crveno svijetlo)
            RESET       C5:0            resetiraj brojač C5
            JMP TO      2               idi na STEP 2
OTHRW
IF              N       I:0/1           čekaj da signal padne u '0'
THEN        JMP TO      3               idi na STEP 3
```

U našem primjeru brojač C5:0 treba nakon pritiska na tipkalo start upaliti crvenu lampicu i početi brojati impulse sa ulaza I:0/1. Kada izbroji svih 10 impulsa gasi se crvena i pali se zelena lampica što znači da je brojač završio sa radom. Pri ponovnom pritisku na tipkalo start ciklus bi se ponovio

U 1 koraku se upisuje konstanta 10 u C5:0/PRE – preset vrijednost brojača

U 2 koraku se aktivira brojač C5:0 na pritisak tipkala start i gasi se zelena lampica ako je bila upaljena iz prošlog ciklusa brojanja, a pali crvena

U 3 koraku se svakim prolazom programa povećava vrijednost brojača C5:0/ACC za 1

U 4 koraku se prati stanje brojača i ako ono skoči u 1 znači da je brojač nabrojio 10 impulsa sa ulaza I:0/1 i može se upaliti zelena,a ugasiti crvena lampica. Ako brojač nije izbrojio svih 10 impulsa program se vraća na korak 3 tek kada impuls padne u 0 da bi program mogao očitati novi impuls.

Vrijednost brojača u primjeru se povećava naredbom INC (increment), ali se isto tako može kombinirati i naredba DEC (decrement) koja smanjuje trenutu vrijednost brojača za 1.

Timeri

Timeri su izlazne naredbe koje nam omogućuju kontrolu nad nekim operacijama u procesu koje su vezane za vrijeme izvođenja. Vrijednosti bitne za rad sa timerima su smještene u timer file, a one se dijele na:

T4:0/EN - Statusni bit timera je bit koji nam govori da li je brojač aktivan('1') ili nije ('0').

T4:0/TT - Statusni bit timera je bit koji nam govori da li brojač radi (ACC<PRE)

T4:0/DN - Statusni bit timera je bit koji nam govori da li je određeni brojač završio sa radom

T4:0/PRE - Preset vrijednost veličine riječi je vrijeme koje mora proteći da bi timer završio sa radom

T4:0/ACC – Vrijednost koja uvijek pokazuje koliko je vremena prošlo od startanja timera

Primjer:

```
STEP 1
IF      NOP                     uvijek aktivno
THEN    RESET   T4:0            resetiraj timer T4
        LOAD    8               upiši konstantu 8 u akumulator
        TO      T4:0/PRE        upiši vrijednost iz akumulatora

STEP 2
IF                              I:0/0   ako je ulaz 0/0 aktivan (tipkalo start)
THEN    SET     T4:0            aktiviraj timer T4
        RESET   O:0/1           ugasi izlaz 0/0 (zeleno svijetlo)
        SET     O:0/1           aktiviraj izlaz 0/1 (crveno svijetlo)

STEP 3
IF                              T4:0/DN ako je prošlo 8 sekundi
THEN    SET     O:0/0           aktiviraj izlaz 0/0 (zeleno svijetlo)
        RESET   O:0/1           ugasi izlaz 0/1 (crveno svijetlo)
        JMP TO  2               idi na STEP 2
```

U našem primjeru pritiskom na tipkalo start treba se aktivirati timer T4:0 i upaliti crvena lampica, a ugasiti zelena ako je bila otprije aktivirana. Nakon 8 sekundi koliko je zadano vrijeme timera crvena lampica se mora ugasiti, a zelena upaliti.

U 1 koraku se upisuje konstanta 8 u T4:0/PRE – preset vrijednost timera

U 2 koraku se aktivira timer T4:0 na pritisak tipkala start i gasi se zelena lampica ako je bila upaljena iz prošlog ciklusa, a pali crvena

U 3 koraku se kontrolira kada je timer T4:0 završio sa radom (T4:0/DN=1 tj isteklo je 8 sekundi) i kada završi pali se zelena lampica, a gasi crvena. Programska kontrola se vraća na drugi korak i čeka na pritisak tipkala start.

Preset vrijednost timera osim u sekundama također možemo zadavati u stotinkama i tisućinkama sekunde samo to treba dodatno naglasiti.

Primjer:

IF		NOP	
THEN	LOAD	400	upiši konstantu 400 u akumulator
	TO	T4:0/PRE	upiši vrijednost iz akumulatora
	WITH	TSC	
THEN	LOAD	66	upiši konstantu 66 u akumulator
	TO	T4:1/PRE	upiši vrijednost iz akumulatora
	WITH	HSC	

U primjeru se preset vrijednost timera T4:0 zadala sa 400 tisućinki sekunde (TSC – thousandth of seconds), dok se preset vrijednost timera T4:1 zadala sa 66 stotinki sekunde (HSC – hundredths of seconds)

Samoodržanje

Često se u praksi za pokretanje i zaustavljanje raznih uređaja (motori, ventili itd) koriste tipkala. Da bi PLC nakon pritiska na tipkalo za startanje to zapamtio i prenio na odgovarajući izlaz potrebno je upisati sljedeći kod:

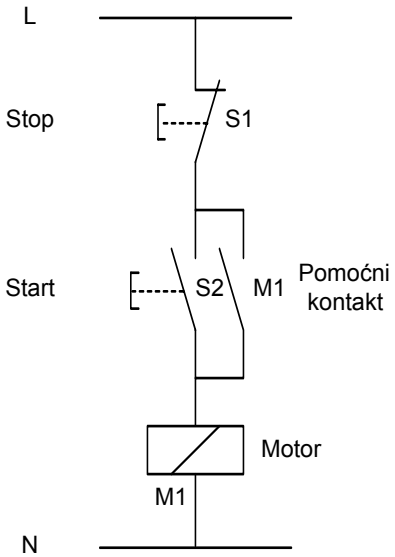
IF			I:0/0	START tipkalo
	OR		O:0/0	motorska sklopka
	AND	N	I:0/1	STOP tipkalo
THEN	=		O:0/0	motorska sklopka

Dakle u trenutku kada se pritisne tipkalo START, a nije pritisnuto tipkalo STOP prema uvijetnom dijelu naredbe uključiti će se motorska sklopka i motor će proraditi. Već u sljedećem prolazu ciklusa rada PLC-a (ms) tipkalo START ne mora biti pritisnuto jer će O:0/0 sam sebe držati uključenim (samoodržanje). Prekid rada motora možemo ostvariti pritiskom na tipkalo STOP. Dobro je primjetiti da se izlazno stanje motora (O:0/0) koristi u uvijetnom dijelu naredbe.

5 Kontaktni dijagrami

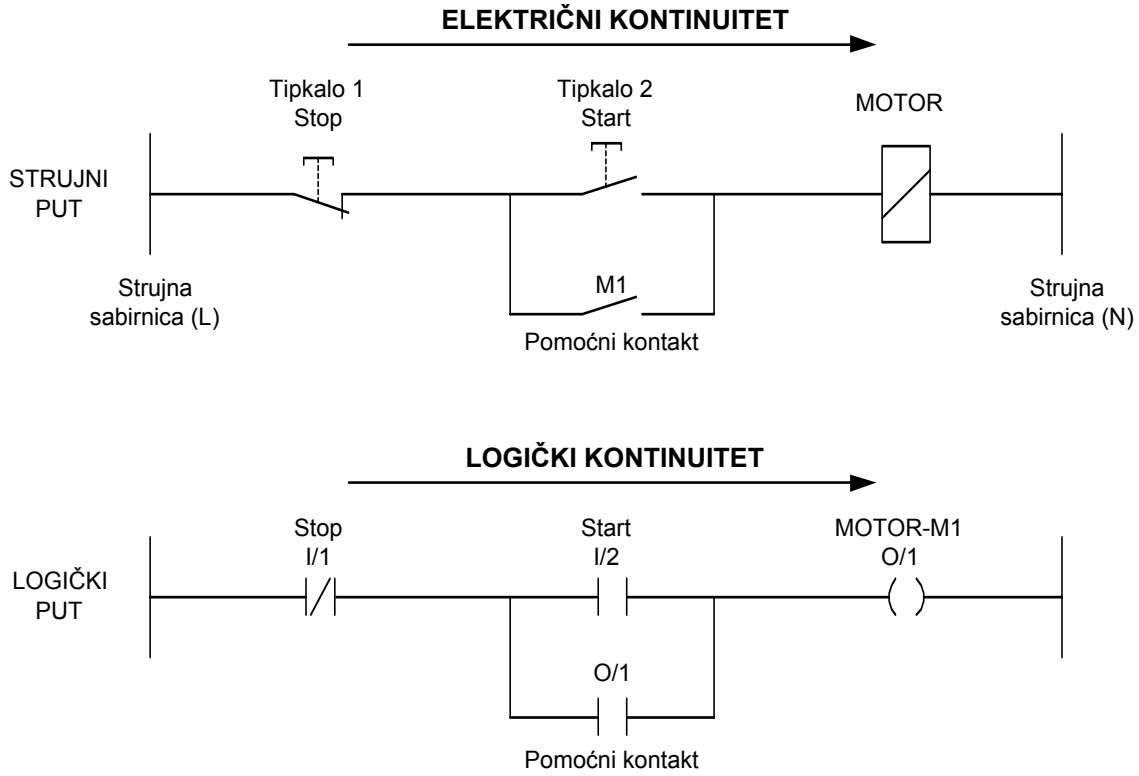
5.1 Opis

Kontaktni dijagrami (ladder diagrami) nastali su na bazi strujnih upravljačkih shema kojima se prikazuje protok struje u strujnom krugu i koje služe električarima kao podloga za ožičenje istog.



Slika – Strujna shema čvrsto ožičenogStart-Stop upravljačkog kruga

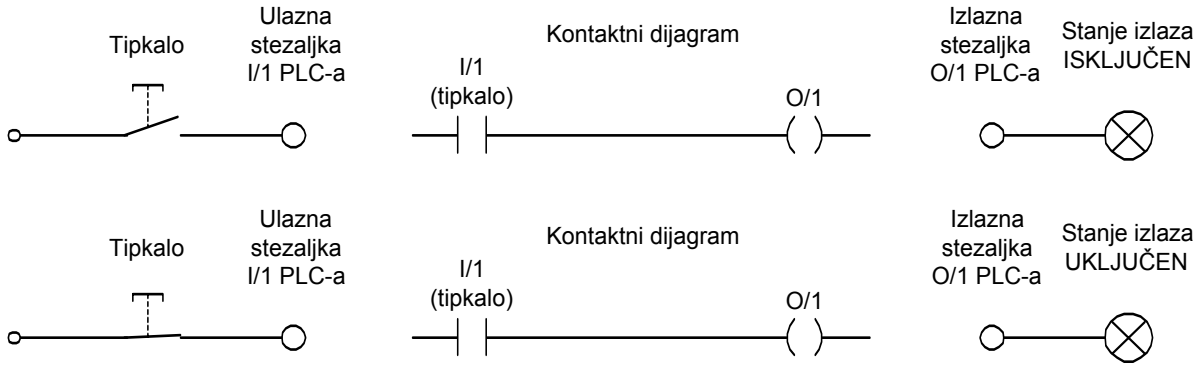
Kako se na slici vidi strujna shema se sastoji od dva horizontalna mrežna vodiča, a struja kroz krug teče odozgo prema dolje. Svaki strujni krug u strujnoj shemi prikazan je kao zaseban strujni put, a svaki strujni put sadrži minimalno jedan upravljani uređaj (npr. motor,relej,žarulja ili slično). Iz strujnog puta se vidi da je rad upravljanog uređaja određen uvjetima(npr. tipkala,pomoćni kontakti i slično) za njegovo uključenje. Kontaktni dijagram PLC programskog jezika vrlo je sličan strujnom putu iz strujne sheme.



Slika – Usporedba strujnog i logičkog puta

Kod strujne sheme simboli predstavljaju stvarne uređaje (kontakte) i njihovo ožičenje, dok kod kontaktnih dijagrama koji koriste slične simbole oni predstavljaju naredbe u programu. Kontaktni dijagram je dio upravljačkog softvera PLC-a za razliku od strujne sheme koja predstavlja stvarni tok struje u strujnom krugu. Još jedna razlika između kontaktnog dijagrama i strujne sheme je u tome što strujna shema prikazuje stanje kontakata (otvoreno ili zatvoreno) dok se u kontaktnom dijagramu ispituje da li je neka naredba istinita '1' ili neistinita '0' što ne mora imati veze sa stanjem kontakata priključenih na ulazne stezaljke PLC uređaja.

Svaki programski logički put u kontaktnom dijagramu mora imati najmanje jednu izlaznu naredbu, a obično sadrži jedan ili više uvijeta koji moraju biti zadovoljeni da bi se izvršila izlazna naredba. Uvjeti (uvjetne naredbe) su najčešće signali koji dolaze sa uređaja priključenih na ulaz PLC-a u kombinaciji sa statusom izlaza, pomoćnih memorijskih varijabli, vremenskih i brojačkih članova. Na desnoj strani svakog logičkog puta nalazi se izlazna naredba koja se aktivira/deaktivira s obzirom na stanje uvjeta. Izlazne naredbe su npr. 'uključi izlaz' naredba koja uključuje npr. izlazni relej PLC uređaja, interne naredbe PLC-a kao manipulacija bitovima, vremenskim i brojačkim članovima te matematičke naredbe. Na slici dan je primjer veze ozmeđu fizičkih kontakata PLC-a i kontaktnog dijagrama.



Slika – Veza između fizičkih kontakata na PLC-u i kontaktnog dijagrama

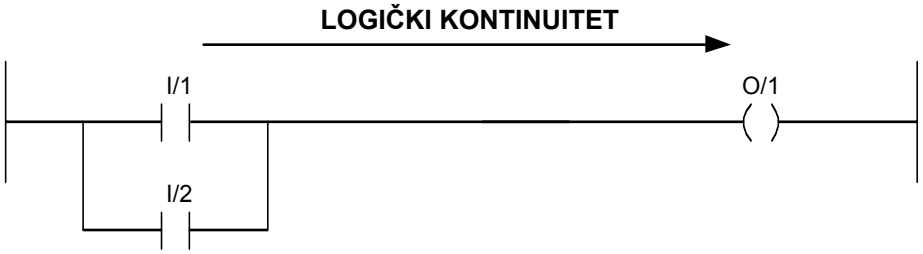
Program će stalno kontrolirati stanje fizičkog ulaza I/1 i prema tome upravljati izlazom O/1. Ovo je najbanalniji primjer kontaktnog dijagrama ali i kod programiranja se u principu koriste samo dvije osnovne kombinacije i njihovi izvodi.

Osnovne kombinacije naredbi čine 'I' i 'ILI' logičke operacije. Kontaktni dijagram serijski povezanih naredbi odnosno 'I' logičke operacije dan je na slici. Primjer ovakve operacije je pokretanje lifta. Da bi se lift pokrenuo moraju biti zadovoljena dva osnovna uvjeta – vrata od lifta su zatvorena i lift nije preopterećen težinom. Ovdje je izlaz O/1 (output 1 – start motora lifta) u logičkom stanju'1' kada su uvijeti I/1 ulaz(input 1 - senzor zatvorenih vrata) i I/2 ulaz (input 2 – senzor preopterećenja lifta) u logičkom stanju '1'. Kada bilo koji od ta dva uvijeta nije ispunjen gubi se logički kontinuitet i lift se neće pokrenuti.

I/1	I/2	O/1
0	0	0
1	0	0
0	1	0
1	1	1

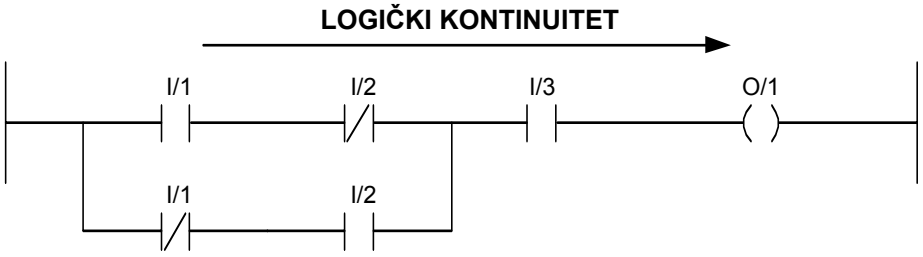
Slika – Serijski povezane naredbe ('I' logička operacija)

Kontaktni dijagram paralelno povezanih naredbi odnosno 'ILI' logičke operacije dan je na slici. Primjer ovakve operacije su automatska vrata na robnim kućama. Vrata će se otvoriti kada bilo koji od dva uvijeta bude ispunjen - ili senzor sa vanjske ili unutarnje strane vrata detektira osobu. Ovdje je izlaz O/1 (output 1 – start motora automatskih vrata) u logičkom stanju'1' kada je uvijet I/1 ulaz (input 1 – senzor sa vanjske strane vrata) ili I/2 ulaz (input 2 – senzor sa unutarnje strane vrata) u logičkom stanju '1'. Kada niti jedan od ta dva uvijeta nije ispunjen gubi se logički kontinuitet i vrata neće biti otvorena.



I/1	I/2	O/1
0	0	0
1	0	1
0	1	1
1	1	1

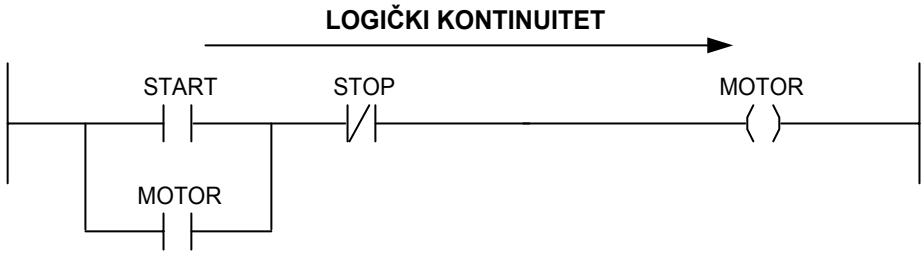
Najčešće se osnovne logičke operacije kombiniraju pa iz njih možemo izvesti kompletnu boleovu algebru. Takav pristup daje nam bezbroj mogućnosti kod samog programiranja. Primjer ovakve kombinacije dan je na slici gdje je prikazana XOR logička funkcija kombinirana sa I logičkom funkcijom.



I/1	I/2	I/3	O/1
0	0	0	0
1	0	0	0
1	1	0	0
1	0	1	1
0	1	0	0
0	1	1	1
1	1	1	0

Programi se sastoje od desetak i više logičkih krugova pa treba naglasiti da se program napisan u kontaktnom dijagramu izvodi odozgo prema dolje tj. od prvog logičkog kruga prema zadnjem.

Često se u praksi za pokretanje i zaustavljanje raznih uređaja (motori, ventili itd) koriste tipkala. Da bi PLC nakon pritiska na tipkalo za startanje to zapamtio i prenio na odgovarajući izlaz potrebno je napisati sljedeći logički krug.



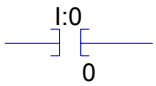
Dakle u trenutku kada se stisne tipkalo START, a nije stisnuto tipkalo STOP prema uvijetnom dijelu logičkog kruga uključiti će se motorska sklopka i MOTOR će proraditi. Već u sljedećem prolazu ciklusa rada PLC-a (ms) tipkalo START ne mora biti pritisnuto jer će MOTOR sam sebe držati uključenim (samoodržanje). Prekid rada motora možemo ostvariti pritiskom na tipkalo STOP. Dobro je primjetiti da se izlazno stanje motora (MOTOR) koristi u logičkoj operaciji odlučivanja.

5.2 Naredbe na razini bita

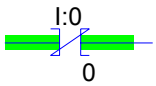
Osnovne naredbe na razini bita i općenito najkorištenije naredbe u PLC programima su naredbe relejnog tipa. Postoje dva tipa relejnih naredbi i to su ulazne (input) i izlazne(output). Ulazne naredbe se koriste da prate/kontroliraju stanje pojedinih bitova bitnih za rad pisanog programa (bitovi ulaznog memorijskog spremnika, bitovi izlaznog memorijskog spremnika, stanja kontrolnih bitova timera i brojača, i sl.), te to stanje unose u program pri svakom ciklusu. Izlazne naredbe su općenito naredbe koje unose dobivenu vrijednost iz svog logičkog puta (runga-kruga) na određenu memorijsku lokaciju (izlazni memorijski spremnik, FIFO stog, memorijske lokacije-markere i sl.). Da bi se naredbe lakše shvatile iza svake naredbe prikazan je kontakt dijagram i dijagram stanje bita – vrijeme.

Osnovne naredbe - tzv. relay type (bit) instructions

Naredba XIC – provjeri da li je zatvoreno (examine if closed)



Naredba XIO – provjeri da li je otvoreno (examine if open)

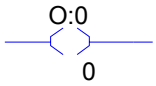


Naredba XIC se koristi kada se želi odrediti da li je adresirani bit u logičkom '1'. Naredba XIO se koristi kada se želi odrediti da li je adresirani bit u logičkom '0'.

Obje naredbe kada se nalaze u logičkom krugu prate status adresiranog bita (ulazne i izlazne stezaljke ili interne memorijske adrese) i prema njegovom stanju propuštaju logički kontinuitet. Primjeri za to mogu biti:

- tipkalo fizički spojeno na adresi I1:0/4
- izlaz spojen na kontrolnu lampicu O0:0/2
- stanje timera T4:3/DN
- stanje bita iz bit filea B3/16

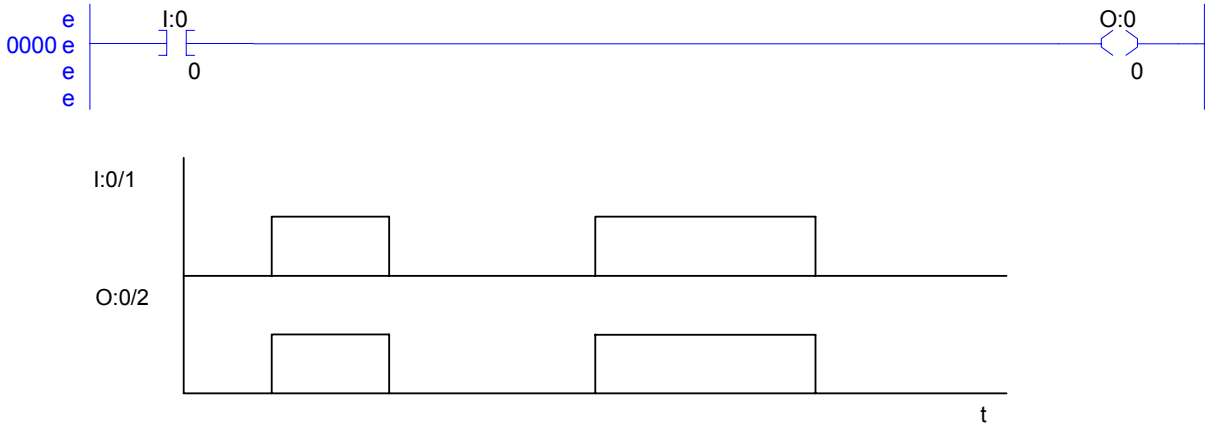
Naredba OTE – uključi izlaz (output energize)



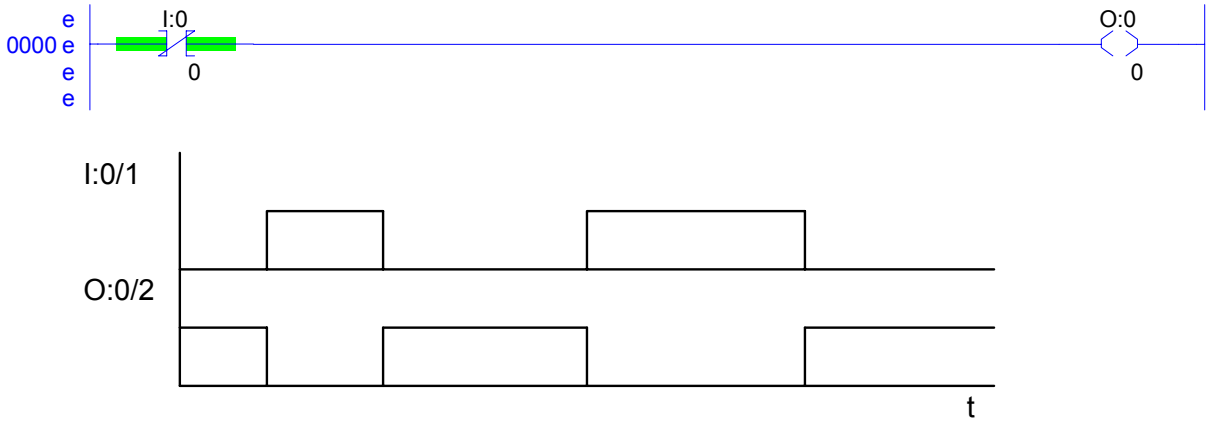
Naredba OTE koristi se da promjeni stanje (0/1) adresirane lokacije veličine bita kada stanje kruga (logički kontinuitet) dođe u '1' odnosno '0'. Primjeri za to su:

- upali lampicu na izlazu O0:0/2
- stavi stanje kruga u B3:5/7

Dijagram stanje – vrijeme za XIC naredbu



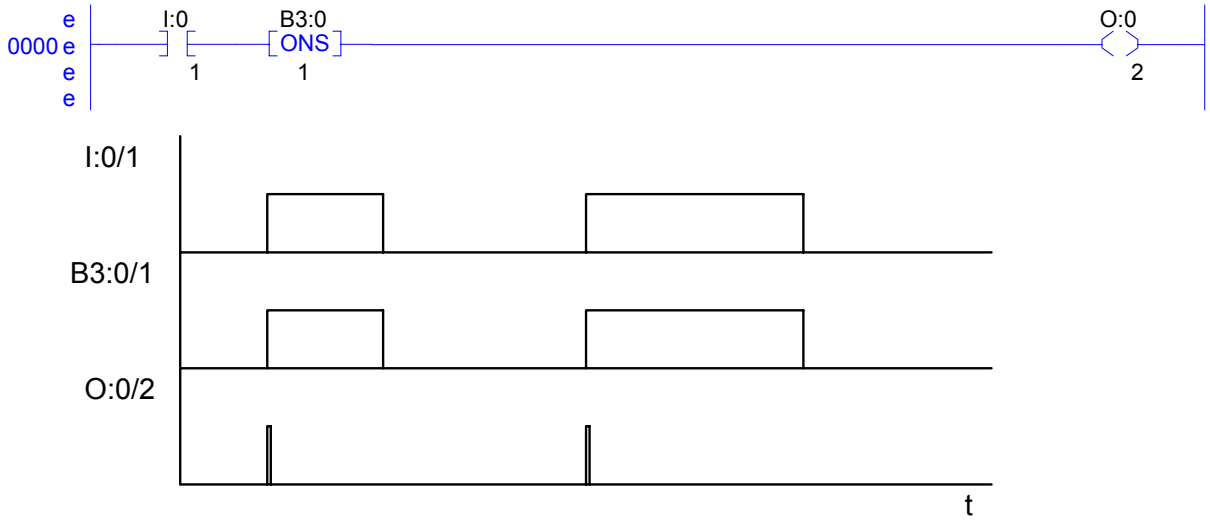
Dijagram stanje – vrijeme za XIO naredbu



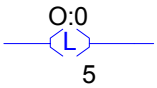
Naredba ONS – jedan prolaz

Naredba ONS (one shot) je ulazna naredba koja kada krug dođe u logičko stanje '1' propušta to stanje samo za jedan ciklus izvođenja programa PLC-a.

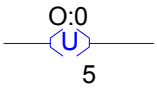
Primjer:



Naredba OTL – setiraj izlaz

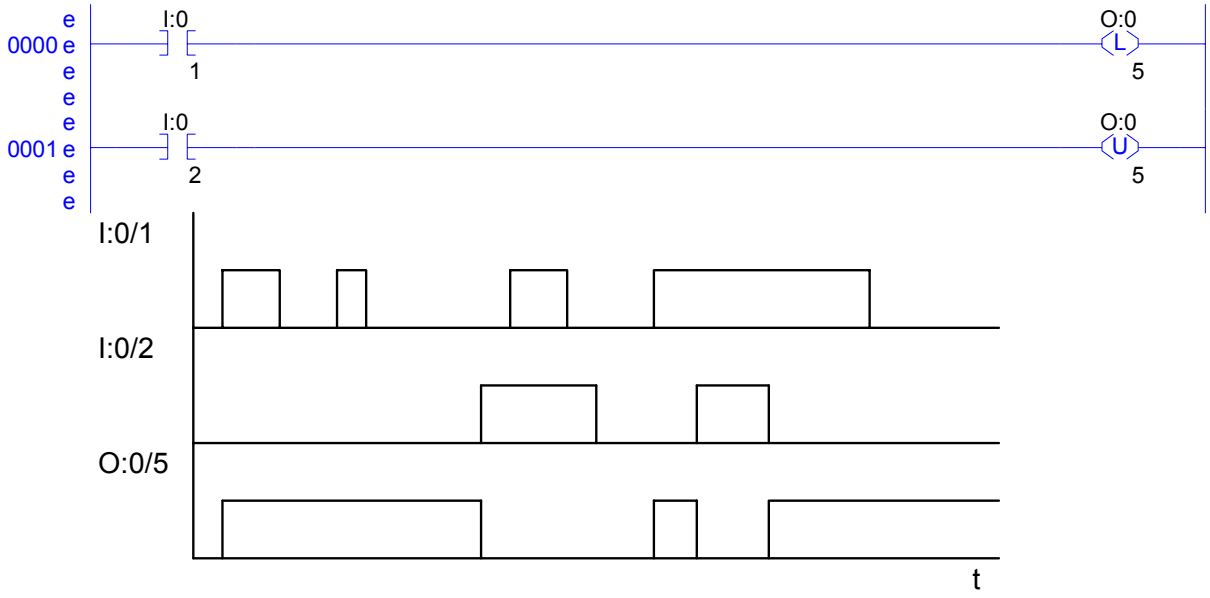


Naredba OTU – resetiraj izlaz



Naredbe OTL (output latch) i OTU (output unlatch) su izlazne naredbe kojima upravljamo stanjem pojedinog bita i uvijek dolaze u paru.

Primjer:



Iz dijagrama stanje-vrijeme na danom primjeru se vidi da će se izlaz O:0/5 uključiti (setirati) kada I:0/1 skoči u 1, ali će ostati uključen i nakon što I:0/1 padne u 0. Izlaz O:0/5 može isključiti (resetirati) samo aktiviranje ulaza I:0/2. Ove dvije naredbe se najčešće koriste kod uporabe tipkala i rada sa procesima gdje treba pamtiti stanja.

5.3 Timeri (timer instructions)

Timeri su izlazne naredbe koje nam omogućuju kontrolu nad nekim operacijama u procesu koje su vezane za vrijeme. Vrijednosti bitne za rad sa timerom su smještene u timer file, a one se dijele na:

- Kontrola i status timera (start timera, rad, kraj rada)

  T4:0/EN – pokazuje stanje kruga u kojem je timer (enable)

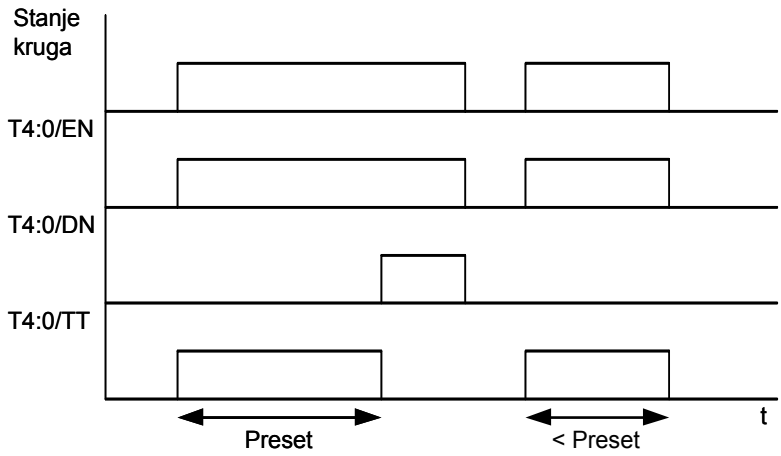
  T4:0/TT – pokazuje kada timer radi (timer timing)

  T4:0/DN – pokazuje kada je timer završio (done)
- T4:0/PRE - Preset vrijednost – vremenska vrijednost kojom se zadaje trajanje rada timera
- T4:0/ACC - Akumulator – vremenska vrijednost koja pokazuje koliko vremena je prošlo od startanja timera. Kada je akumulator>=preset timer je završio sa radom

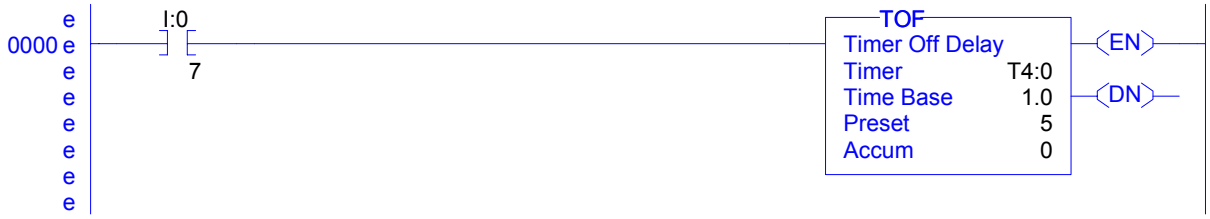
5.3.1 Naredba TON – timer, on-delay

TON timer počinje brojati vrijeme kada je stanje kruga dođe u visoko. Sve dok je stanje kruga visoko, vrijednost akumulatora se povećava. Kada vrijednost akumulatora dostigne preset vrijednost timer je završio sa radom. Akumulator se resetira kada stanje kruga dođe u nisko.

Dijagram stanje – vrijeme za TON

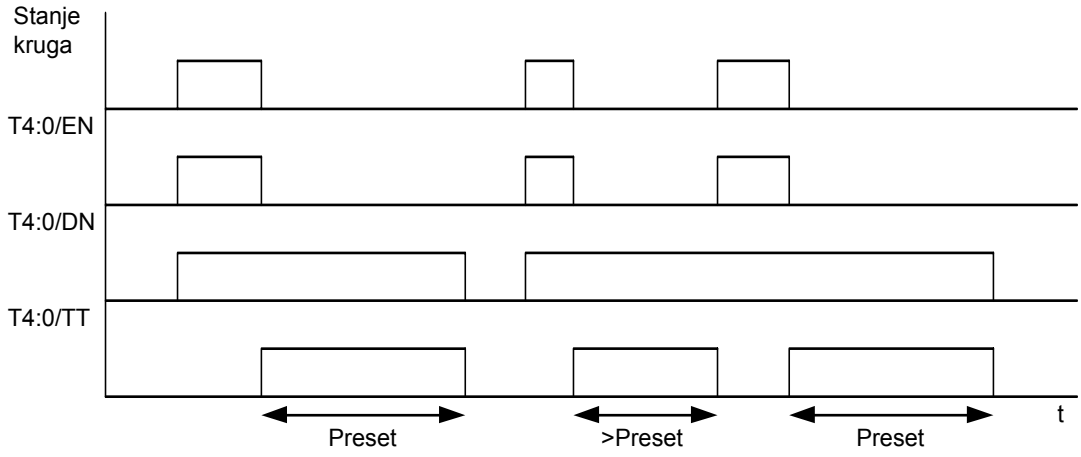


5.3.2 Naredba TOF – timer, of-delay



TOF timer počinje brojati vrijeme kada je stanje kruga dođe u nisko. Sve dok je stanje kruga nisko, vrijednost akumulatora se povećava. Kada vrijednost akumulatora dostigne preset vrijednost timer je završio sa radom. Akumulator se resetira kada stanje kruga dođe u visoko.

Dijagram stanje – vrijeme za TOF



5.4 Brojači (counter instructions)

Brojači su izlazne naredbe koje nam omogućuju kontrolu nad nekim operacijama u procesu koje su vezane za odbrojavanje. Vrijednosti bitne za rad sa brojačima su smještene u counter file, a one se dijele na

- Kontrola i status brojača (start brojača, rad, kraj rada)

  C5:0/CU – pokazuje stanje kruga u kojem je brojač za brojanje unaprijed (count up enable bit)

  C5:0/CD – pokazuje stanje kruga u kojem je brojač za brojanje unazad (count down enable bit)

  C5:0/DN – pokazuje kada je brojač završio (done)

  C5:0/OV – pokazuje kada je brojač došao do maksimalne vrijednosti do koje može brojati (32767) – coun overflow bit

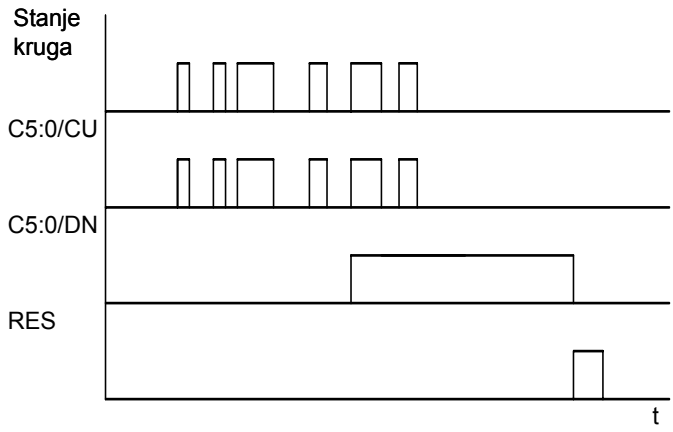
  C5:0/UN – pokazuje kada je brojač došao do minimalne vrijednosti do koje može brojati (-32767) – coun underflow bit

- C5:0/PRE - Preset vrijednost – vrijednost koju brojač treba izbrojati (od -32767 do 32767)
- C5:0/ACC - Akumulator – vrijednost koja pokazuje koliko je brojač izbrojio impulsa. Kada je akumulator>=preset brojač je završio sa radom

5.4.1 Naredba CTU – Count up

Svaki put kada krug u kojem se nalazi brojač CTU prijeđe iz niskog u visoko stanje vrijednost akumulatora se poveća za 1. Brojanje započinje od vrijednosti 0 i može ići do vrijednosti 32767 (2^{15}). Kada vrijednost akumulatora dosegne preset vrijednost brojač je završio sa radom. Akumulator se može resetirati samo naredbom RES (reset).

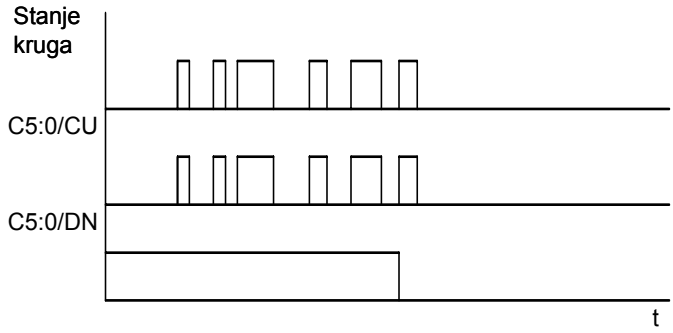
Dijagram stanje – vrijeme za CTU (Primjer kada je preset=5)



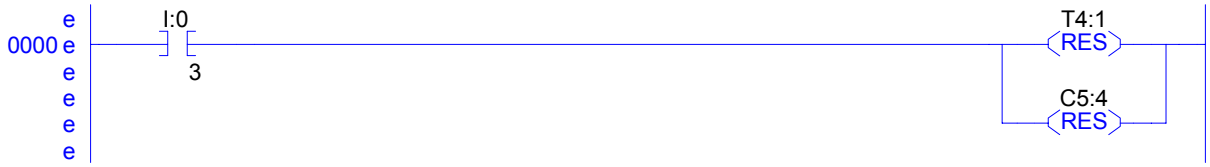
5.4.2 Naredba CTD – Count down

Svaki put kada krug u kojem se nalazi brojač CTD prijeđe iz niskog u visoko stanje vrijednost akumulatora se poveća za -1. Brojanje započinje od vrijednosti 0 i može ići do vrijednosti -32767 (2^{15}). Kada vrijednost akumulatora dosegne vrijednost preset+1 brojač je završio sa radom.

Dijagram stanje – vrijeme za CTU (Primjer kada je preset=5)



5.4.3 Naredba RES – reset

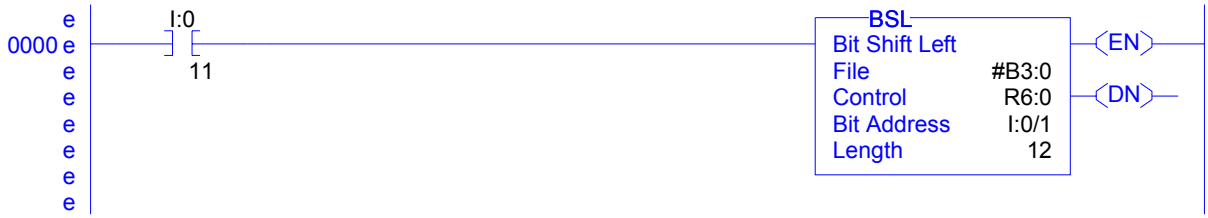


Naredba RES je izlazna naredba koja resetira tj. postavlja na početnu vrijednost timere i brojače. Kada krug u kojem je naredba RES dođe u stanje visoko resetira se timer ili brojač koji je definiran (slika). Resetiranje će postići efekt tek kada krug dođe u stanje nisko. U tablici su dani elementi timera i brojača na koje naredba RES djeluje.

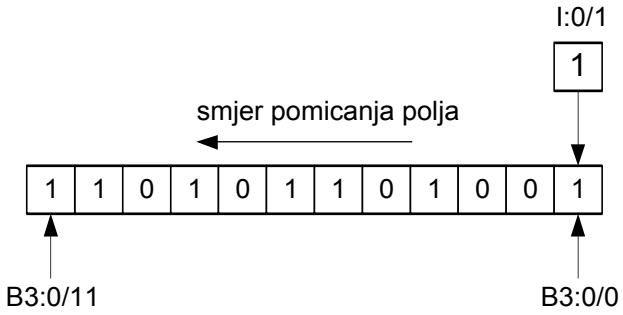
Elementi timera	Elementi brojača
ACC vrijednost u 0	ACC vrijednost u 0
T4:0/DN	C5:0/DN
T4:0/TT	C5:0/OV
T4:0/EN	C5:0/UN
	C5:0/CU
	C5:0/CD

5.5 Naredbe pretvorbe (conversion instructions)

Naredba BSL – pomicanje polja uljevo



Naredba BSL (bit shift left) je izlazna naredba koja svaki puta na promjenu stanja logičkog kruga iz niskog u visoko upiše vrijednost bita sa adrese I:0/1 (bit addres) u polje bitova na adresi B3:0 (file) i pomakne polje za jedan bit uljevo. Vrijednost 'length' nam govori koliko je veliko polje bitova (u našem primjeru od B3:0/0 do B3:0/11). Ovakvo polje bita se naziva lijevi shift registar.



Control file R6:0 je file u kojem se nalaze kontrolni bitovi:

R6:0/EN – govori nam kad je logički krug u stanju visoko

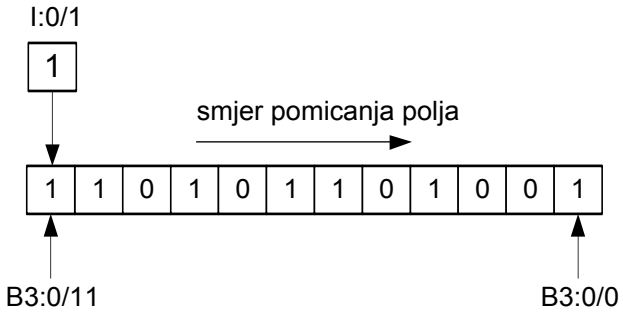
R6:0/DN – govori nam kad se polje pomaklo za jedan bit uljevo

R6:0/UL – govori nam kada je prvi bit došao do zadnje pozicije

Naredba BSR – pomicanje polja udesno

Naredba BSL (bit shift rigth) je izlazna naredba koja svaki puta na promjenu stanja logičkog kruga iz niskog u visoko upiše vrijednost bita sa adrese I:0/1 (bit addres) u polje

bitova na adresi B3:0 (file) i pomakne polje za jedan bit udesno. Vrijednost 'length' nam govori koliko je veliko polje bitova (u našem primjeru od B3:0/11 do B3:0/0). Ovakvo polje bita se naziva desni shift registar.



Control file R6:0 je file u kojem se nalaze kontrolni bitovi:

R6:0/EN – govori nam kad je logički krug u stanju visoko

R6:0/DN – govori nam kad se polje pomaklo za jedan bit uljevo

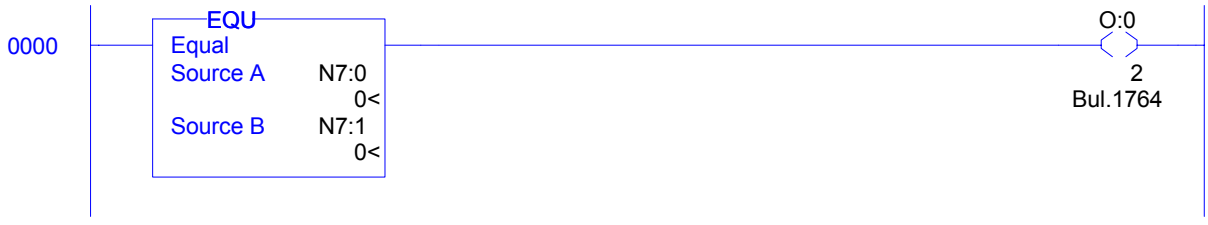
R6:0/UL – govori nam kada je prvi bit došao do zadnje pozicije

5.6 Naredbe na razini riječi

Kao i naredbe na razini bita, naredbe na razini riječi mogu biti ulazne ili izlazne. Kao rezultat ulaznih naredbi uvije ćemo dobivati informaciju veličine bita koja se šalje u logički krug. Rezultat izlaznih naredbi će biti informacija veličine riječi (16 ili 32 bita). Adrese N7:0, N7:1 itd uzete su samo kao primjer jer se podatak veličine 16 bitne riječi sprema u npr N7 integer file.

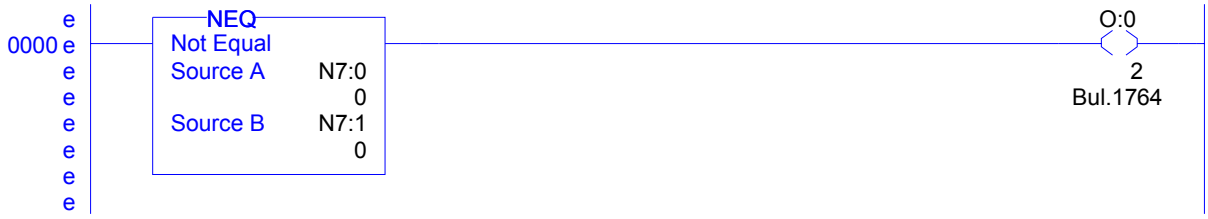
5.6.1 Naredbe usporedbe (Compare instructions)

Naredba EQU – Equal



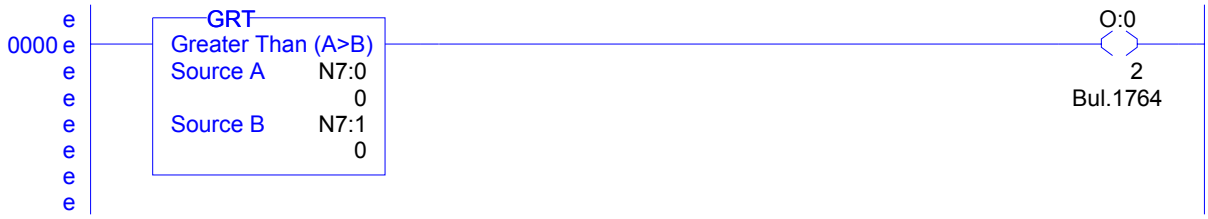
Naredba EQU je ulazna naredba koja uspoređuje jednakost podatka veličine riječi sa adrese N7:0 (source A) i podatka sa adrese N7:1 (source B). Ako su vrijednosti jednake EQU u logički krug šalje '1'. Obe zadane vrijednosti (source A i source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba NEQ – Not equal



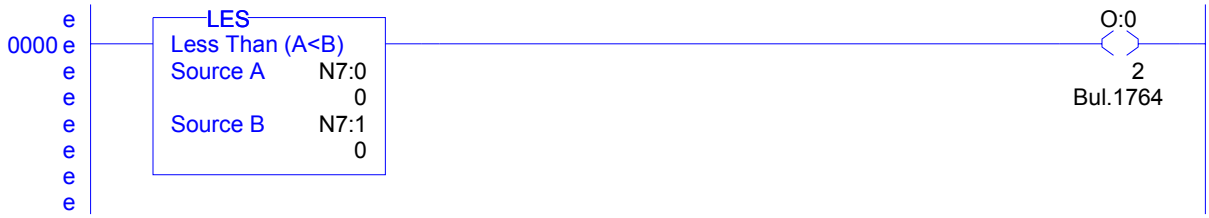
Naredba NEQ je ulazna naredba koja uspoređuje nejednakost podatka veličine riječi sa adrese N7:0 (source A) i podatka sa adrese N7:1 (source B). Ako su vrijednosti nejednake NEQ u logički krug šalje '1'. Obe zadane vrijednosti (source A i source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba GRT – Greather than



Naredba GRT je ulazna naredba koja uspoređuje da li je vrijednost podatka veličine riječi sa adrese N7:0 (source A) veća od vrijednosti sa adrese N7:1 (source B). Ako je vrijednost podatka A veća od vrijednosti podatka B naredba GRT šalje '1' u logički krug. Obe zadane vrijednosti (source A i source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba LES – Less than

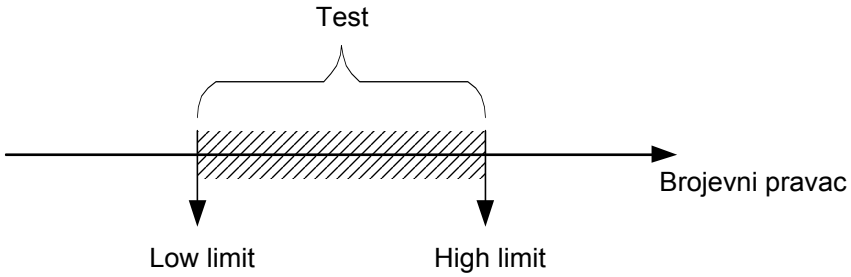


Naredba LES je ulazna naredba koja uspoređuje da li je vrijednost podatka veličine riječi sa adrese N7:0 (source A) manja od vrijednosti sa adrese N7:1 (source B). Ako je vrijednost podatka A manja od vrijednosti podatka B naredba GRT šalje '1' u logički krug. Obe zadane vrijednosti (source A i source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

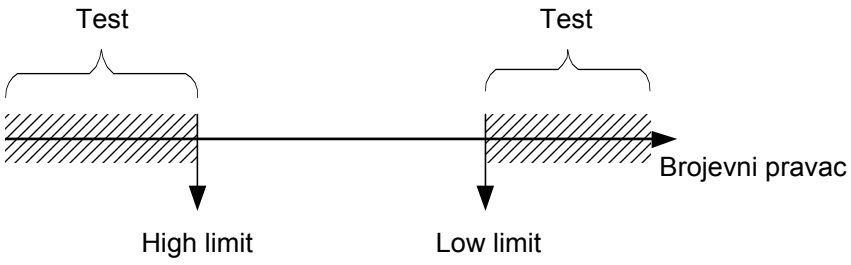
Naredba LIM – Limit test

Naredba LIM je ulazna naredba koja uspoređuje da li se vrijednost podatka veličine riječi sa adrese N7:1 (Test) nalazi između granica koje su zadane na adresama N7:0 (Low limit) i N7:0 (High limit). Postoje dva moguća slučaja:

Ako je High limit vrijednost > Low limit vrijednosti i ako se Test vrijednost nalazi između High limit i Low limit onda će naredba LIM će davati '1' u logički krug (vidi sliku).



Ako je High limit vrijednost < Low limit vrijednosti i ako se Test vrijednost ne nalazi između High limit i Low limit onda će naredba LIM će davati '1' u logički krug.



Sve tri zadane vrijednosti (Test, High limit, Low limit) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

5.6.2 Matematičke naredbe (Math instructions)

Kod matematičkih naredbi treba naglasiti da ako se u postupku računanja desi da se dijeli sa 0 ili prijeđe maksimalan dozvoljeni broj (word - ± 32767 ili long word - ± 2147483647) PLC će javiti grešku.

Naredba ADD – zbrajanje

Naredba ADD je izlazna naredba koja zbraja vrijednost podatka veličine riječi na adresi N7:0 (source A) sa vrijednošću na adresi N7:1 (source B) i rezultat sprema na adresu N7:2 (dest – destination). Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa. Ako zbrojena vrijednost prelazi veličinu registra u koji se sprema PLC će javiti grešku.

Naredba Sub – oduzimanje

Naredba SUB (subtract) je izlazna naredba koja oduzima vrijednost podatka veličine riječi na adresi N7:1 (source B) od vrijednosti na adresi N7:1 (source B) i rezultat sprema na adresu N7:2 (dest – destination). Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa. Ako vrijednost 'Dest' prelazi veličinu registra u koji se sprema PLC će javiti grešku.

Naredba MUL – množenje

Naredba MUL (multiply) je izlazna naredba koja množi vrijednost podatka veličine riječi na adresi N7:0 (source A) sa vrijednošću na adresi N7:1 (source B) i rezultat sprema na adresu N7:2 (dest – destination). Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa. Ako umnožak (Dest) prelazi veličinu registra u koji se sprema PLC će javiti grešku.

Naredba DIV – dijeljenje

Naredba DIV (divide) je izlazna naredba koja dijeli vrijednost podatka veličine riječi na adresi N7:0 (source A) sa vrijednošću na adresi N7:1 (source A) i rezultat sprema na adresu N7:2 (dest – destination). Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa. Treba naglasiti da ako PLC nema float file on će podijeliti samo cijele brojeve,a ostatak će spremiti u posebni registar (npr. 26 / 4 kao rezultat daće 6, a ostatak 2 će spremiti u za to previđen registar).

Naredba NEG – negativna vrijednost

Naredba NEG (negate) je izlazna naredba koja uzima vrijednost podatka veličine riječi sa adrese N7:0 (source) i potom ju spremi na adresu N7:1 (dest) sa promjenjenim predznakom. Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba SCP – linearna aproksimacija

Naredba SCP (scale with parameters) je izlazna naredba koja vrši linearnu aproksimaciju nad podacima veličine riječi po poznatoj jednadžbi:

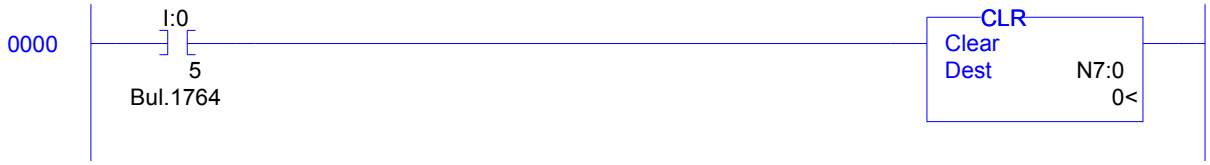
$$Y=(Y_1-Y_0 / X_1-X_0)(X-X_0) + Y_0$$

Gdje su:

Input I:2.0	X
Input min N7:1	X_0
Input max N7:2	X_1
Scaled min N7:3	Y_0
Scaled max N7:4	Y_1
Output O:1.0	Y

Zadane vrijednosti moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

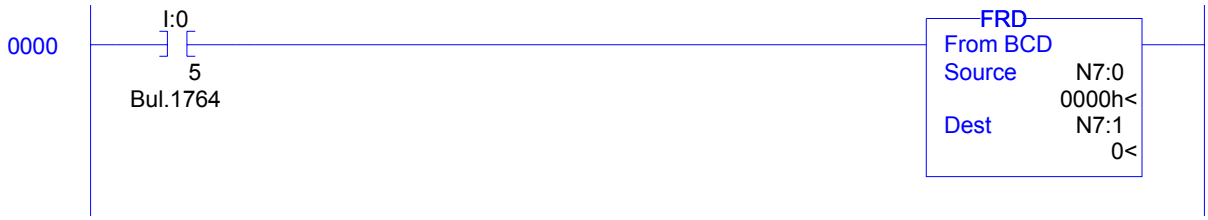
Naredba CLR – brisanje podatka



Naredba CLR (clear) je izlazna naredba koja briše podatak bilo koje veličine sa adrese N7:0 (dest)., odnosno postavlja sve bitove sa te adrese u '0'.

5.6.3 Naredbe pretvorbe (conversion instructions)

Naredba FRD – pretvorba iz BCD kod-a



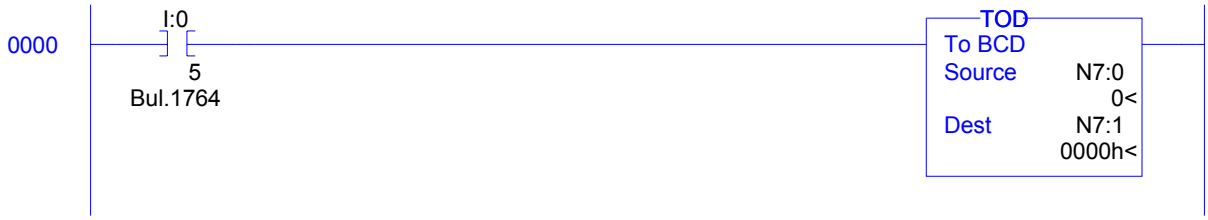
Naredba FRD (from BCD – Binary Coded Decimal) je izlazna naredba koja uzima vrijednost podatka veličine riječi sa adrese N7:0 (source) napisanu u BCD kod-u i pretvara je u njenu binarnu vrijednost te rezultat sprema na adresu N7:1 (dest).

Primjer:

BCD	2673	0010 0110 0111 0011
Dec/Binarno	2673	0000101001110001

Treba naglasiti da ako je veličina izlazne riječi (dest) 16 bita onda ulazna riječ (source) može imati maksimalnu BCD vrijednost 9999.

Naredba TOD – pretvorba u BCD kod



Naredba TOD (to BCD – Binary Coded Decimal) je izlazna naredba koja uzima vrijednost podatka veličine riječi sa adrese N7:0 (source) napisanu u decimalnom ili binarnom obliku u i pretvara je u njenu BCD vrijednost te rezultat sprema na adresu N7:1 (dest).

Primjer:

Dec/Binarno	3547	0000110111011011
BCD	3547	0011 0101 0100 0111

5.6.4 Logičke naredbe (logical instructions)

Logičke naredbe su naredbe vezane za Boleovu algebru samo za podatke veličine riječi.

AND – logičko 'I'

Naredba AND je izlazna naredba koja vrši logički finkciju 'I' nad vrijednostima podataka veličine riječi sa adresa N7:0 (source A) i N7:1 (source B) te rezultat sprema na adresu N7:2 (dest – destination).

Source A (N7:0)	1001 0110 0100 0101
Source B (N7:1)	0011 1100 1011 0100
Destination (N7:2)	0001 0100 0000 0100

Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba OR – logičko 'ILI'

Naredba OR je izlazna naredba koja vrši logički finkciju 'ILI' nad vrijednostima podataka veličine riječi sa adresa N7:0 (source A) i N7:1 (source B) te rezultat sprema na adresu N7:2 (dest – destination).

Source A (N7:0)	1001 0110 0100 0101
Source B (N7:1)	0011 1100 1011 0100
Destination (N7:2)	1011 1110 1111 0101

Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

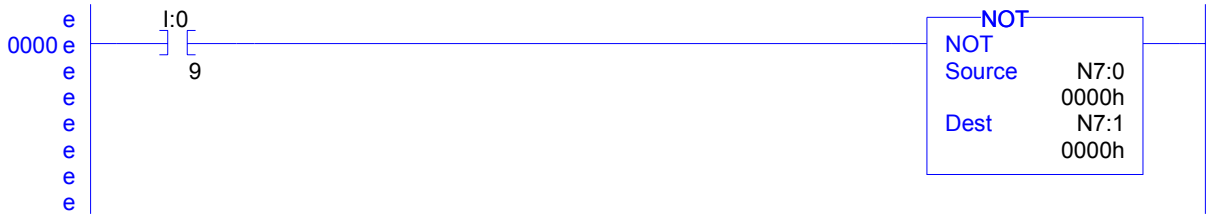
Naredba XOR – logičko ekskluzivno 'ILI'

Naredba XOR je izlazna naredba koja vrši logički finkciju ekskluzivno 'ILI' nad vrijednostima podataka veličine riječi sa adresa N7:0 (source A) i N7:1 (source B) te rezultat sprema na adresu N7:2 (dest – destination).

Source A (N7:0)	1001 0110 0100 0101
Source B (N7:1)	0011 1100 1011 0100
Destination (N7:2)	1010 1010 1111 0001

Zadane vrijednosti (source A, source B) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba NOT – logički 'NE'



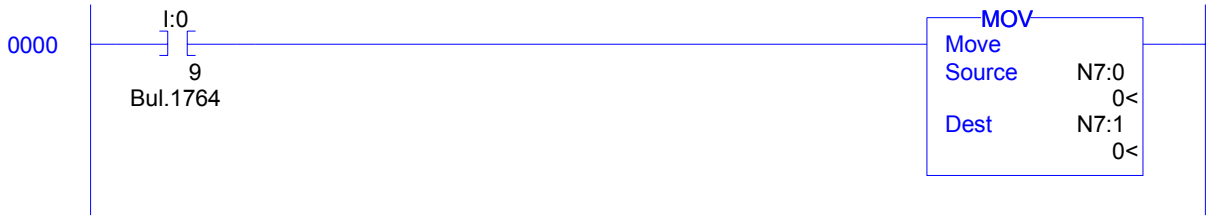
Naredba NOT je izlazna naredba koja uzima vrijednost veličine riječi sa adrese N7:0 (source) i na adresu N7:1 (destination) stavlja njezin jedinični komplement.

Source A (N7:0)	1001 0110 0100 0101
Destination (N7:1)	0110 1001 1011 1010

Veličine memorijskih registara (source, dest) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

5.7 Naredbe nad podacima (data instruction)

Naredba MOV – premještanje podatka



Naredba MOV je izlazna naredba koja premješta vrijednost podataka veličine riječi sa adrese N7:0 (source) na adresu N7:1 (destination). Veličine memorijskih registara (source, dest) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba MVM – premještanje kroz masku

Naredba MVM je izlazna naredba koja vrijednost podatka veličine riječi sa adrese N7:0 (source) premješta na adresu N7:2 (destinstion) filtrirajući ga kroz vrijednost na adresi N7:1 (mask).

Primjer:

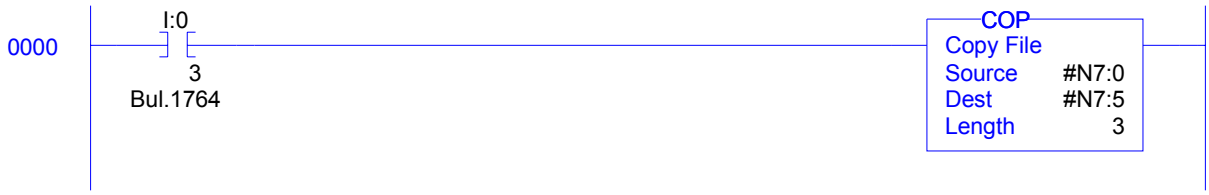
Source (N7:0)	1001 0110 0100 0101
Mask (N7:1)	0011 1100 1011 0100
Destination (N7:2)	0001 0100 0000 0100

Dakle samo na onim adresama gdje je bit maske jednak 1 informacija iz N7:0 će se prenesti u N7:2. Ako se vrijednost iz maske promjeni iz 1 u 0 a prije toga je prenjela informaciju 1 iz N7:0 u N7:2 informacija ostaje u N7:2 nepromjenjena odnosno 1. Uzmimo gornji primjer i promjenimo vrijednost maske N7:1.

Source (N7:0)	1001 0110 0100 0101
Mask (N7:1)	1011 1000 1100 0001
Destination (N7:2)	1001 0100 0100 0101

Veličine memorijskih registara (source, mask, dest) moraju biti iste veličine (16,32 bita) i mogu se mijenjati za vrijeme izvođenja programa.

Naredba COP – jednostruko kopiranje



Naredba COP (copy) je izlazna naredba koja kopira vrijednost podatka veličine riječi sa adrese N7:0 (source) na adresu N7:1 (dest). Vrijednost 'length' nam govori koliko riječi iza riječi N7:0 će se također kopirati – sa slike 'length'=3 i prema tome: N7:0 se kopira u N7:5

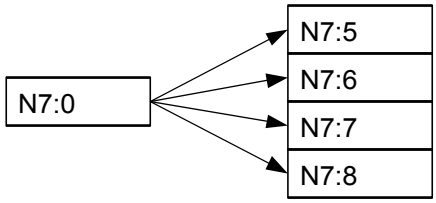
N7:1 se kopira u N7:6

N7:2 se kopira u N7:7

N7:3 se kopira u N7:8

Naredba FLL – višestruko kopiranje

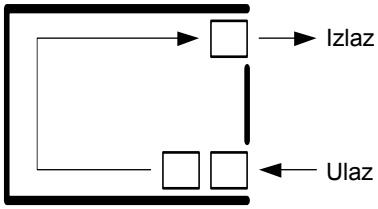
Naredba FLL (fill file) je izlazna naredba koja uzima vrijednost podatka veličine riječi sa adrese N7:0 (source) i kopira ga na više adresa počevši od adrese N7:5.



Vrijednost 'length' nam govori na koliko će se riječi iza riječi N7:5 kopirati vrijednost iz N7:0 (slika - 3).

5.8 FIFO memorijski registar

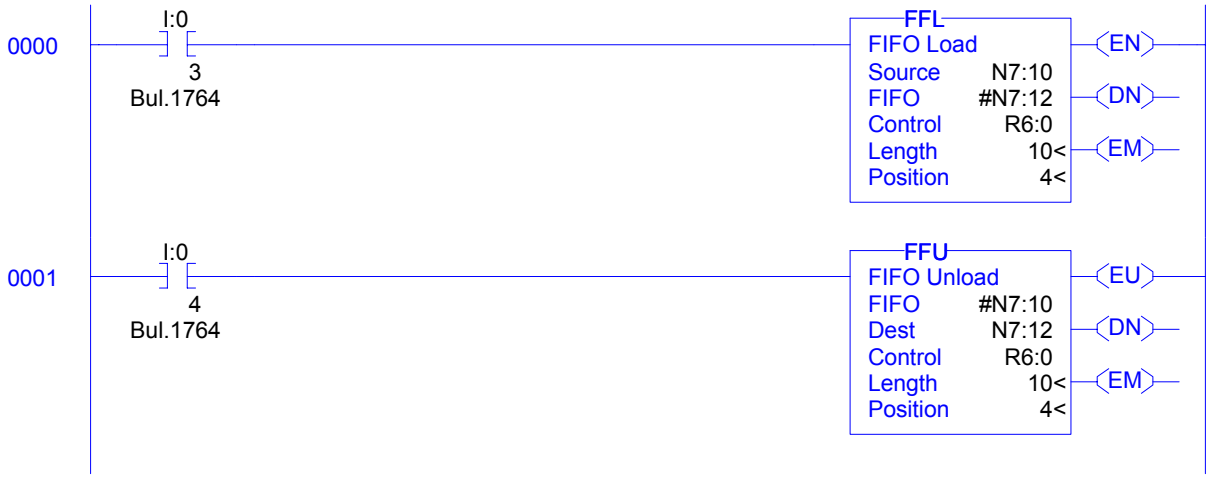
FIFO memorijski registar (first in, first out – prvi unutra, prvi van) je memorijsko polje podataka veličine riječi koje radi na principu ulazno/izlaznog skladišta (inf. stog).



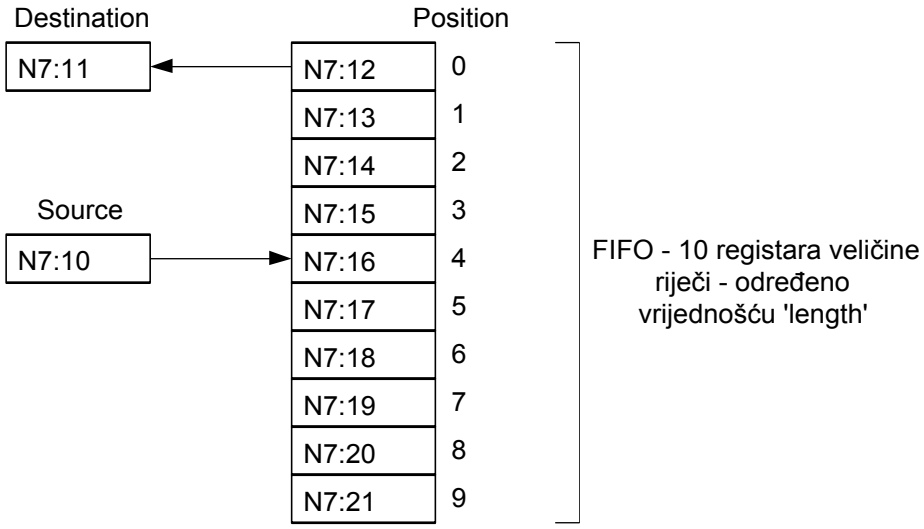
Kako je prikazano na slici podatak koji prvi uđe u registar prvi i izlazi iz njega. Za korištenje FIFO memorijskog registra najčešće se upotrebljavaju FFL i FFU naredbe zajedno.

Naredba FFL – FIFO load

Naredba FFU – FIFO unload



Korištenje FIFO registra pokazaćemo na danom primjeru.



Adresa #N7:12 određuje početnu adresu stoga, a vrijednost 'length' broj registara iza adrese N7:12 koje ćemo koristiti (veličina stoga).

Kada logički krug u kojem se nalazi FFL naredba dođe iz stanja nisko u stanje visoko u registar N7:16 će se upisati trenutačno stanje vrijednosti podatka sa adrese N7:10 (source). Pri tome će vrijednost position koja je brojem 4 određivala N7:16 povećati se za 1. Sljedeći put kada logički krug u kojem se nalazi FFL naredba dođe iz stanja nisko u stanje visoko vrijednost position je 5 te će se vrijednost iz N7:10 će se upisati u N7:17. FIFO registar se na taj način može popunjavati sve do length-1=position kada će se popuniti zadnji slobodni FIFO registar.

Kada logički krug u kojem se nalazi FFU naredba dođe iz stanja nisko u stanje visoko u registar N7:11 (destination) će se upisati stanje vrijednosti podatka sa adrese N7:12, tj podatak koji je prvi ušao u stog prvi izlazi van. Pri tome se vrijednost position smanjila za 1 i sve vrijednosti u stogu su se pomakle za jednu adresu niže – tako na adresu N7:12 dolazi podatak sa adrese N7:13 i pri sljedećoj promjeni stanja logičkog kruga FFU naredbe iz niskog u visoko stanje ta vrijednost ide u N7:11 (destination).

Control file R6:0 je file u kojem se nalaze kontrolni bitovi:

R6:0/EN – govori nam kada je FFL logički krug u stanju visoko

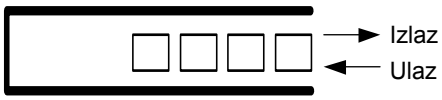
R6:0/EU – govori nam kada je FFU logički krug u stanju visoko

R6:0/DN – govori nam kad je stog pun

R6:0/EM – govori nam kada je stog prazan

5.9 LIFO memorijski registar

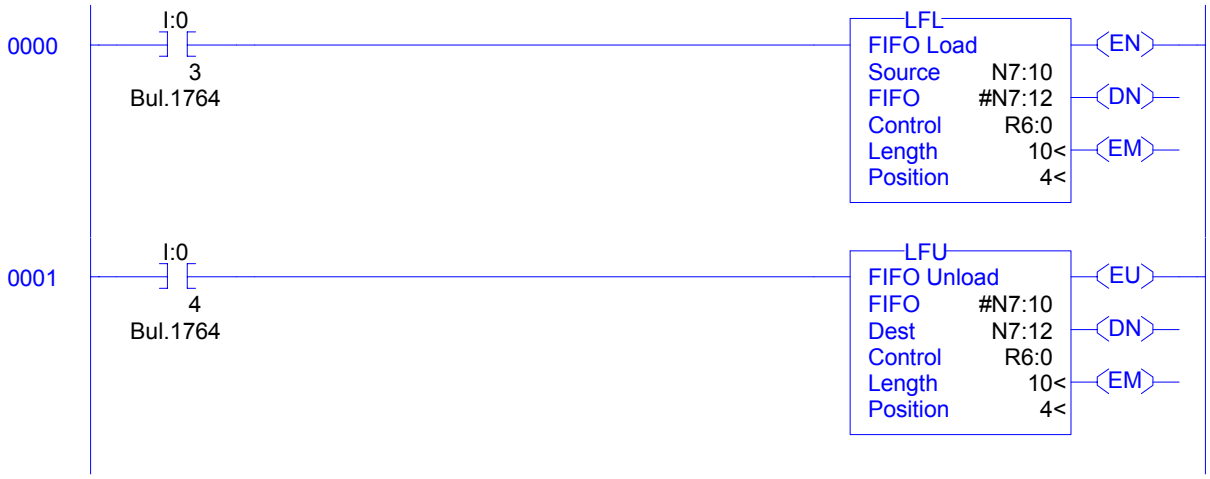
LIFO memorijski registar (last in, first out – zadnji unutra, prvi van) je memorijsko polje podataka veličine riječi koje radi na principu ulazno/izlaznog skladišta (inf. stog).



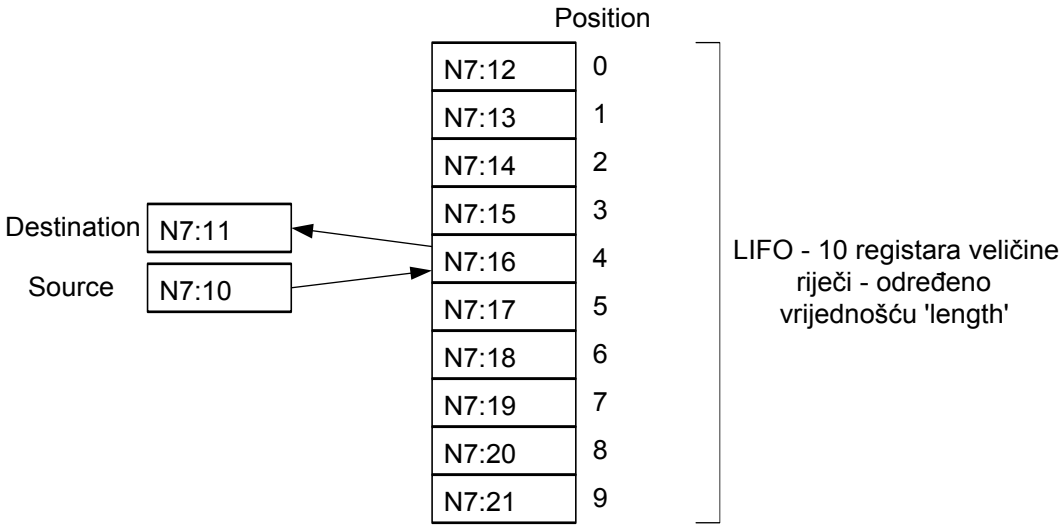
Kako je prikazano na slici podatak koji prvi uđe u registar zadnji izlazi iz njega. Za korištenje LIFO memorijskog registra najčešće se upotrebljavaju LFL i LFU naredbe zajedno.

Naredba LFL – LIFO load

Naredba LFU – LIFO unload



Korištenje LIFO registra pokazaćemo na danom primjeru.



Adresa #N7:12 određuje početnu adresu stoga, a vrijednost 'length' broj registara iza adrese N7:12 koje ćemo koristiti (veličina stoga).

Kada logički krug u kojem se nalazi LFL naredba dođe iz stanja nisko u stanje visoko u registar N7:16 će se upisati trenutačno stanje vrijednosti podatka sa adrese N7:10 (source). Pri tome će vrijednost position koja je brojem 4 određivala N7:16 povećati se za 1. Sljedeći put kada logički krug u kojem se nalazi LFL naredba dođe iz stanja nisko u stanje visoko vrijednost position je 5 te će se vrijednost iz N7:10 će se upisati u N7:17. FIFO registar se na taj način može popunjavati sve do length-1=position kada će se popuniti zadnji slobodni FIFO registar.

Kada logički krug u kojem se nalazi LFU naredba dođe iz stanja nisko u stanje visoko u registar N7:11 (destination) će se upisati stanje vrijednosti podatka sa adrese N7:16, tj podatak koji je zadnji ušao u stog prvi izlazi van. Pri tome se vrijednost position smanjila za 1 i pri sljedećoj promjeni logičkog kruga LFU naredbe iz niskog u visoko stanje u registar N7:11 (destination) bi se upisala vrijednost iz N7:15.

Control file R6:0 je file u kojem se nalaze kontrolni bitovi:

R6:0/EN – govori nam kada je LFL logički krug u stanju visoko

R6:0/EU – govori nam kada je LFU logički krug u stanju visoko

R6:0/DN – govori nam kad je stog pun

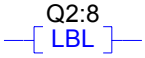
R6:0/EM – govori nam kada je stog prazan

5.10 Naredbe usmjerivanja programa (program control instructions)

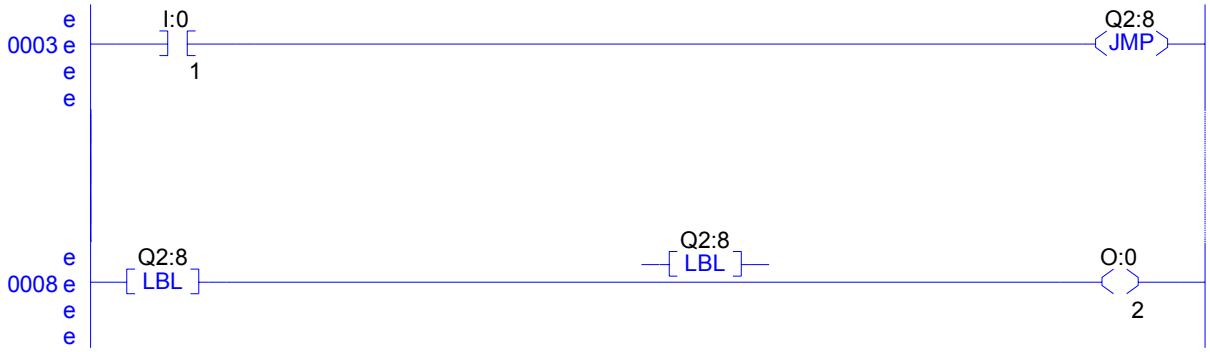
Naredbe usmjeravanja programa se koriste za promjenu redosljeda izvršavanja krugova kontakt dijagrama. Također se može program prusmjeriti na potprograme (subrutine). Te naredbe se uglavnom koriste da bi se smanjili vrijeme trajanja programa i poboljšala efikasnost.

Naredba JMP – skoči na označen krug

Naredba LBL – oznaka kruga

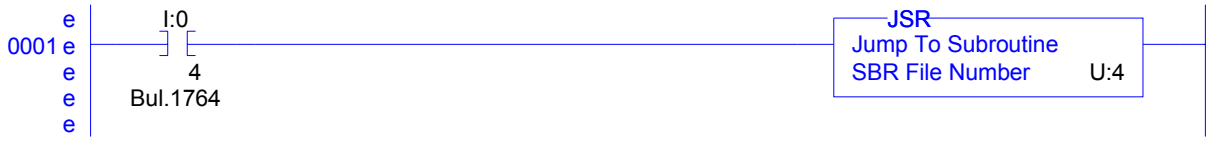


Naredba JMP (jump) i LBL (label) uvijek koristimo u paru pa ih je najbolje objasniti na slici:

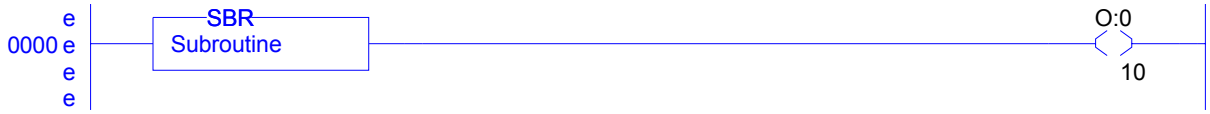


Kada se na ulaznoj stezaljki I:0/1 pojavi signal logički krug 3 (rung 3) dođe u stanje visoko i naredba JMP prebaci kontrolu na rung 8 jer je on označen naredbom LBL. Sve dok je I:0/1 u visokom stanju logički krugovi 4 – 7 se neće izvršavati. Broj 2 pored slova Q na oznaci nam govori da se radi o program file br. 2.

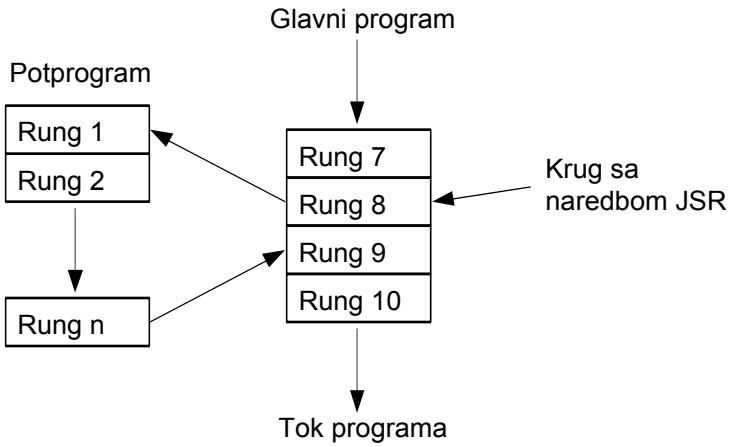
Naredba JSR – skoči na potprogram



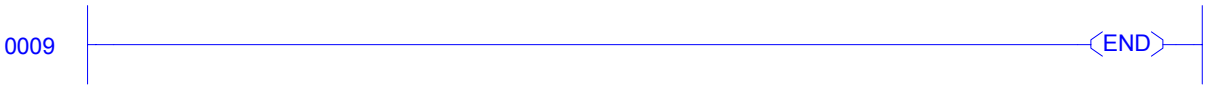
Naredba SBR – oznaka potprograma



Naredba JSR (jump to subroutine) je izlazna naredba koja kada krug u kojem se nalazi dođe u stanje visoko prebacuje kontrolu izvođenja programa na potprogram (ovdje potprogram br. 4). Kada se potprogram izvede do kraja kontrola izvođenja programa nas vraća u glavni program i nastavlja sa radom tamo gdje je stala (slika). Da bi se potprogram mogao izvoditi potrebno je da prva ulazna naredba u njemi bude SBR odnosno oznaka potprograma (subroutine label).



Naredba END – kraj programa



Naredba END je izlazna naredba koja se obavezno nalazi na kraju svakog programa.

6 Rad s računalom

6.1 Vježba 1: Konfiguriranje komunikacije PLC-PC računalo

U ovoj vježbi, upoznati ćemo vas sa RSLinx programskim paketom za komunikaciju između PC računala i PLC-a. Trebati ćete:

- Pokrenuti RSLinx aplikaciju.
- Automatski konfigurirati (engl. Auto-Configure) RSLinx RS-232 driver.

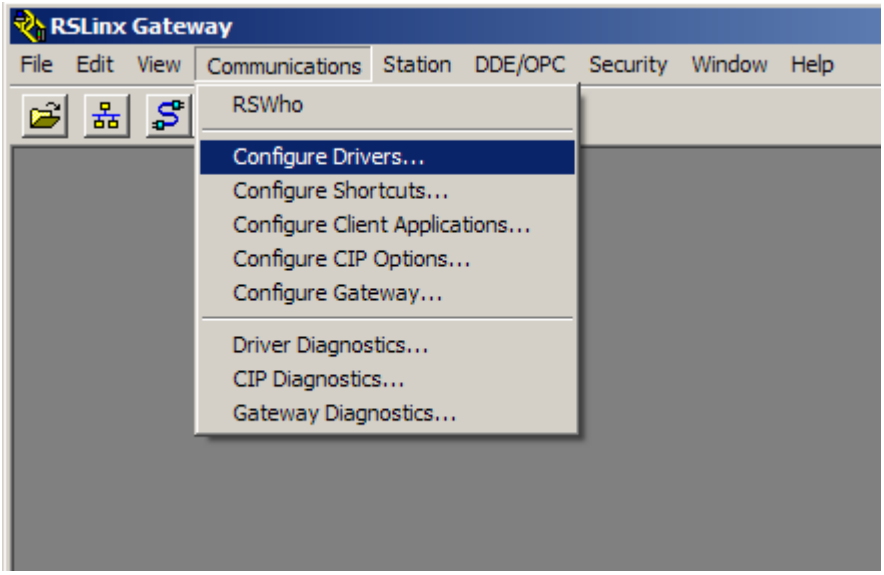
Slijedite dolje navedene korake da bi završili vježbu 1.

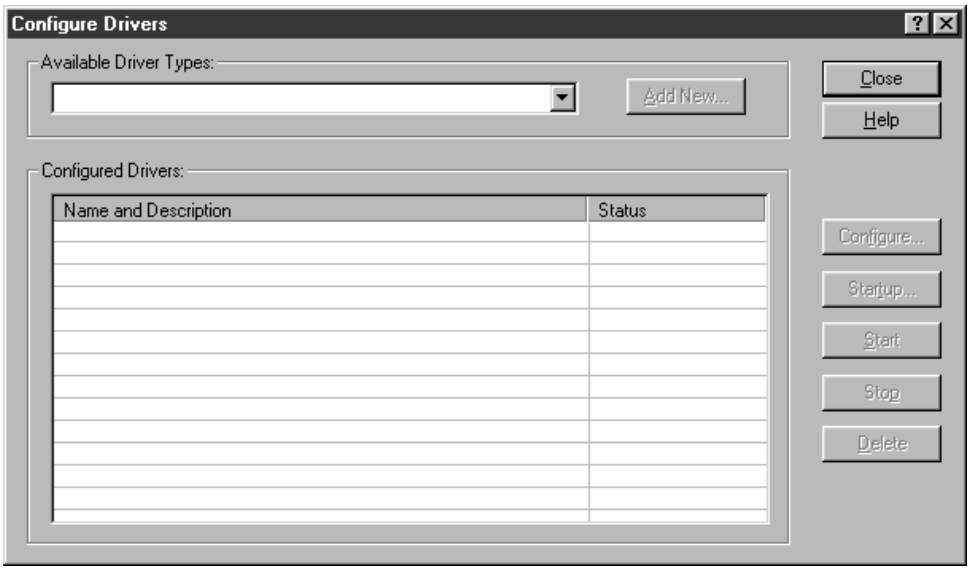
RSLinx je komunikacijski paket koji omogućava PLC-u da komunicira s vašim računalom pokrečući RSLogix 500 software. Prvo moramo konfigurirati način na koji želimo da naše računalo komunicira sa PLC-om.

1. Iz Windows start menia odaberite RSLinx.

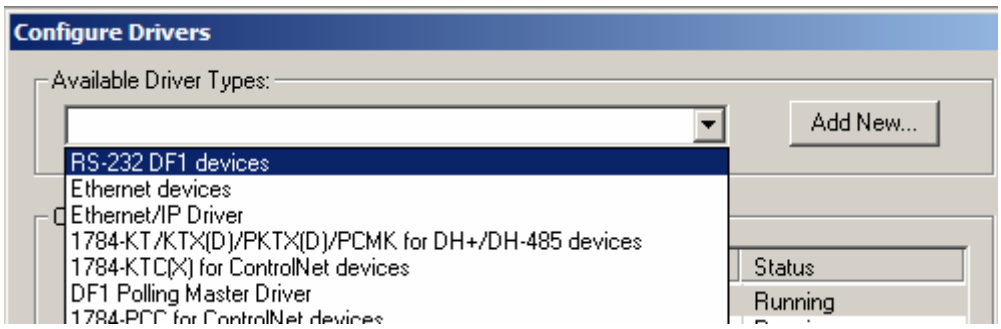
Start →Programs → Rockwell Software → RSLinx → RSLinx

2. Jednom kada se RSLinx pokrene idite na '**Communications**' otvorite njegov meni i odaberite'**Configure Drivers'**.

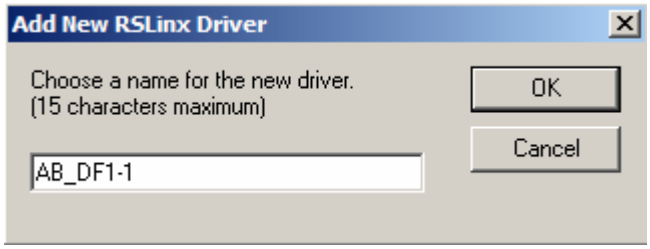




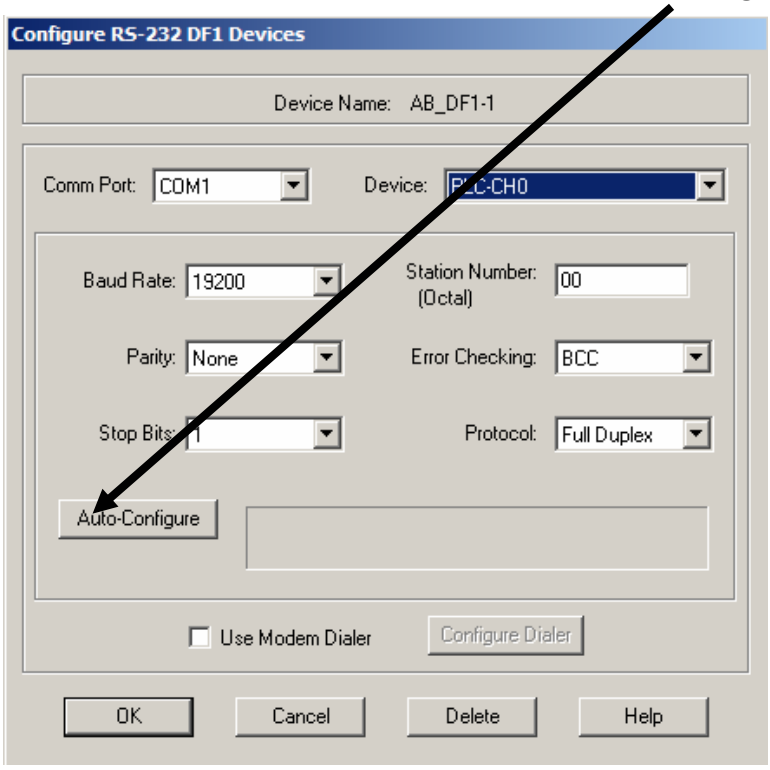
3. Iz **Available Driver Types** menia, izaberite prvi na listi koji je **RS-232 DF1.** Ako su i neki drugi konfiguracijski driveri navedeni ispod "Configured Drivers" možete ih izbrisati. Ako je driver već pokrenut zato što vaše računalo već komunicira sa kontrolerom, zaustavite driver prije nego što ga izbrišete.



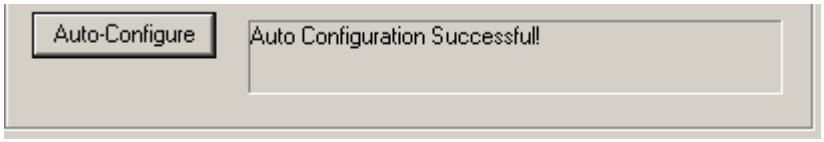
4. Kliknite na **Add New**.
5. Kada se od vas traži - kliknite na **OK**; da bi prihvatili ime po defaultu (engl. **default name**).



6. Tada ćete vidjeti konfiguracijski zaslon za komunikaciju. Ovdje se nalazi mnogo informacija, to može izgledati zbunjujuće, međutim velika značajka RSLinx-a je da automatski može konfigurirati sve parametre za vas pretpostavljajući da ste već spojeni sa vašim kontrolerom. Jednostavno kliknite na **Auto-Configure** tipku.



Jednom kada je RSLinx uspješno testiran i provjeren vidjeti ćete sljedeću poruku u konfiguracijskom prozoru drivera (engl. Driver Configuration) prozoru.



Ako Auto-Configure (automatsko konfiguriranje) nije uspješno, provjerite da li je komunikacijski kabel vašeg računala (PC-a) ispravno spojen sa vašim PLC-om.

7. Kliknite **OK** kako bi izabrali “Driver Configuration” prozor (konfiguracijski prozor drivera).
8. Kliknite **Close** kako bi izabrali “Configure Drivers” prozor.

I to je to!

Završili smo konfiguriranje našeg komunikacijskog drivera, i više to nećemo morati raditi na ovom računalu.

9. Minimizirajte ali ne zatvarajte RSLinx tako da kliknete na u gornjem desnom kutu RSLinx prozora.

6.2 Vježba 2: Izrada novog projekta (New Project)

6.2.1 O ovoj vježbi

U ovoj vježbi ćemo vas uvesti u osnove programiranja PLC-a, koristeći kontrolere iz MicroLogix skupine proizvoda. U ovoj vježbi ćete trebati:

- Izraditi novi projekt.
- Napisati 3 logička kruga (rung) ladder logike, pokrenuti i zaustaviti simulirani motor
- Snimiti (pohraniti) vaš projekt na hard drive (tvrdi disk) PC-a.
- Prebaciti (download) projekt na MicroLogix kontroler na vašoj radnoj stanici (work station).
- Pratite i testirajte vaš program on-line s MicroLogix kontrolerom.

Slijedite dolje navedene korake, kako bi završili vježbu 2.

6.2.2 O MicroLogix kontrolerima

6.2.2.1 MicroLogix

Temelji se na arhitekturi na tržištu vodeće SLC 500 skupine kontrolera McroLogix skupina programabilnih kontrolera omogućava 3 razine kontrole. Mali po veličini, ali jako dobrih performansi, MicroLogix 1000 nudi kontrolne sposobnosti u isplativom, jezgrovitom paketu. MicroLogix 1200 je dovoljno malen da stane u iznimno male prostore, ali dovoljno snažan da osigura širok pojas aplikacija. Dizajniran je da se ekspandira/proširuje po potrebi, MicroLogix 1500 pomaže vam da ostvarite visok nivo kontrole u raznolikosti aplikacija. Ovi mali kontroleri imaju velike brzine, moćne instrukcije i fleksibilnu komunikaciju za aplikacije koje traže: cijena - učinkovitost rješenja.

MicroLogix kontroleri se koriste u širokom spektru aplikacija od malih-velikih brzina, samostalni kontroleri za lokalnu kontrolu, do mrežnih integriranih kontrolera koji omogućuju sistem-razina rješenja, i čak daleki terminalni djelovi koji kombiniraju lokalnu kontrolu i prikupljene podatke sa master kontrolerima koji nadziru složene arhitekture. Ovi mali kontroleri nalaze se u mnogo oblika, a bitne karakteristike su im:

- Velike procesorske brzine, sa tipičnim programskim skeniranjem od 1 do 3 ms po 1K korisničkog programa.
- ugrađeni ulazi i izlazi velikih brzina za:
    - Brojač velikih brzina za obradu signala visokih frekvencija.
    - Ulazi za otkrivanje kratkotrajih pulseva, nezavisnih od ciklusa programa.
    - I/O prekidi za događaje procesirane u realnom vremenu, PTO (pulse train output) ili PWM (pulse width modulation).
    - Podesni filteri za filtriranje signala.
- Prošireni I/O (dostupno samo kod MicroLogix 1200 i MicroLogix 1500).
    - Omogućava kontroleru da se prilagodi tijekom vremena potrebama okoline.
    - Dopušta fleksibilnost u miješanju različitih I/O tipova, da bi se dostigle potrebe aplikacije.

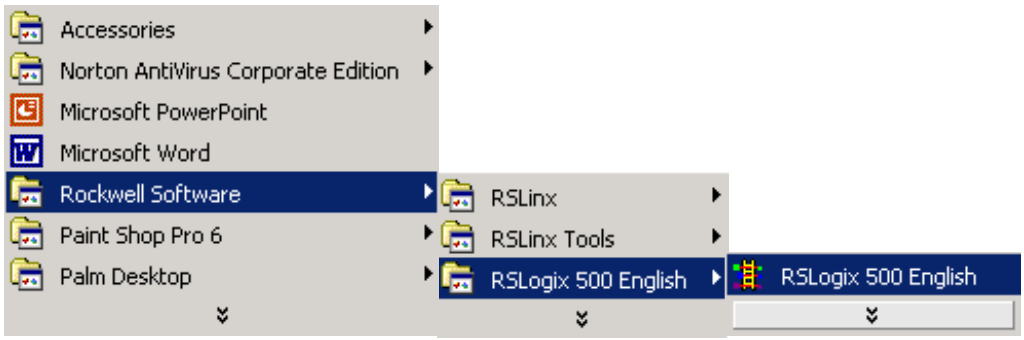
  - Dopušta kontroleru da adresira I/O module, koji se nalaze udaljeni od kontrolera.
- Omogućuje da se mnogostruke opcije protokola integriraju u postojeći kontrolni okoliš ili za izgrađenje efikasnog kontroler - kontroler sistema široke komunikacijeske mreže.
- Progremski softver RSLogix 500.
- I karakteristike koje se jednostavno koriste, kao što je ručno programiranje (samo MicroLogix 1000), trimerski potenciometri (samo MicroLogix 1200 i 1500), premjestanje memorijskih modula u radnom stranju (samo MicroLogix 1200 i 1500), i Data Access Tool (samo MicroLogix 1500).

6.2.3 Pokretanje RSLogix 500 programskog paketa

U ovom djelu vježbe, pokrenuti ćete RSLogix 500 aplikaciju, koji će vam omogućiti da programirate vaš MicroLogix kontroler.

1. **Sa zaslona vašeg računala pokrenite RSLogix 500:**

   Start>Programs>Rockwell Software>RSLogix 500 English>RSLogix 500 English.



6.2.4 Kreiranje: New Controller Project

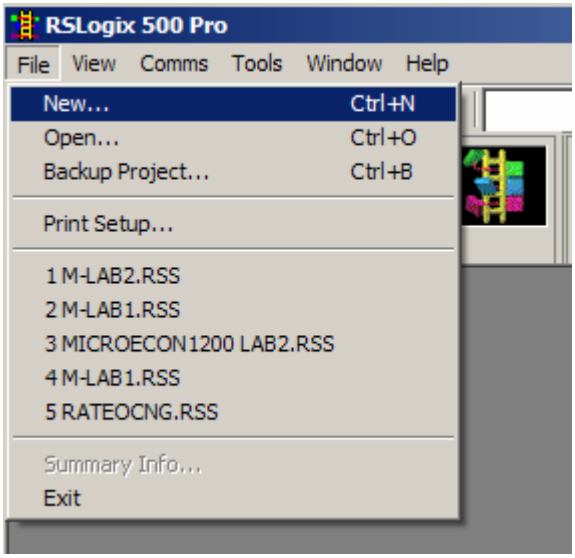
U ovom djelu vježbe, kreirati ćete offline program za vaš MicroLogix kontroler.

Budite sigurni da ste odabrali MicroLogix kontroler koji se nalazi na vašoj radnoj stanici (engl. lab station).

Da bi identificirali koji tip MicroLogix kontrolera je instaliran na vašoj radnoj stanici, jednostavno pogledajte prednji dio vašeg kontrolera i vidjeti ćete da li se radi o MicroLogix 1200, ili MicroLogix 1500 kontroleru.

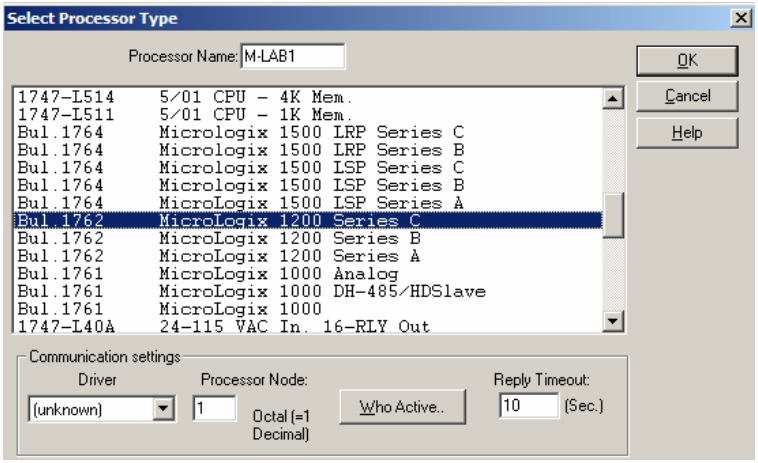
1. **Povećajte RSLogix 500 prozor, tako da kliknete na ikonu maximize u gornjem desnom kutu RSLogix 500 prozora.**

2. Otvorite File meni i odaberite **NEW** da bi kreirali novi dokument (File).

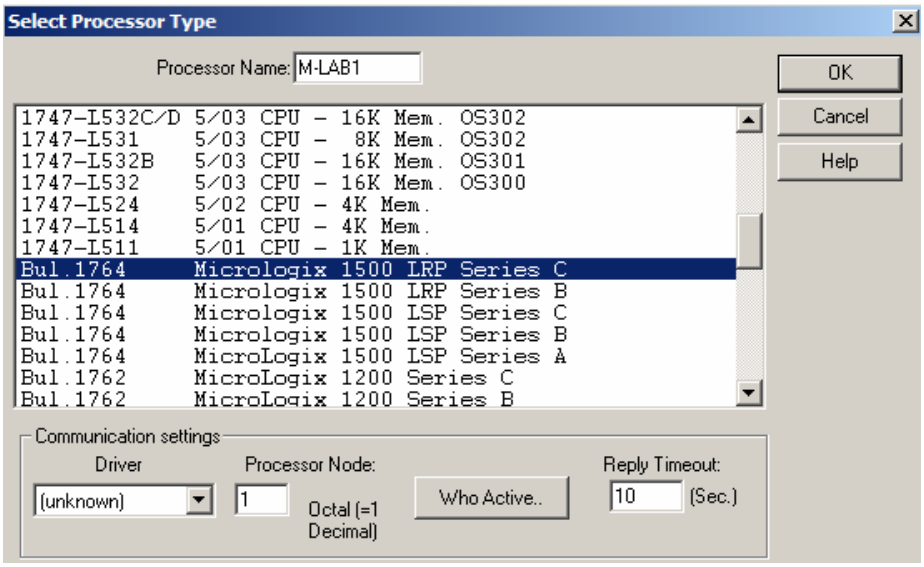


3. Unesite Processor Name: `'M-vježba1'`.
4. Koristeći strelicu za dolje odaberite MicroLogix kontroler koji ćete naći na vašoj radnoj stanici.

Ako se na vašoj radnoj stanici nalazi MicroLogix 1200, odaberite kako je prikazano dolje:



Ako se na vašoj radnoj stanici nalazi MicroLogix 1500, odaberite kako je prikazano dolje:

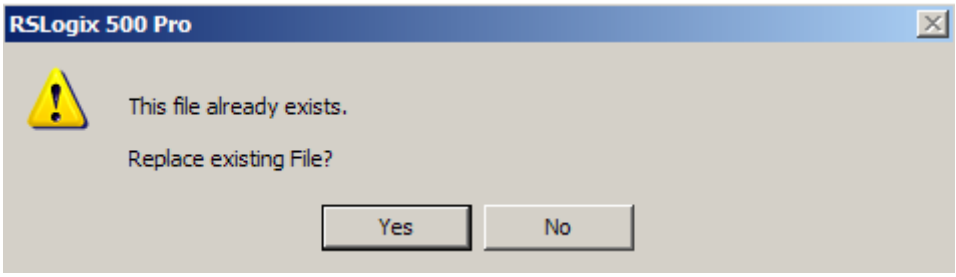


Pitajte instruktora ili asistenta ako niste sigurni koji hardver se nalazi na vašoj radnoj stanici .

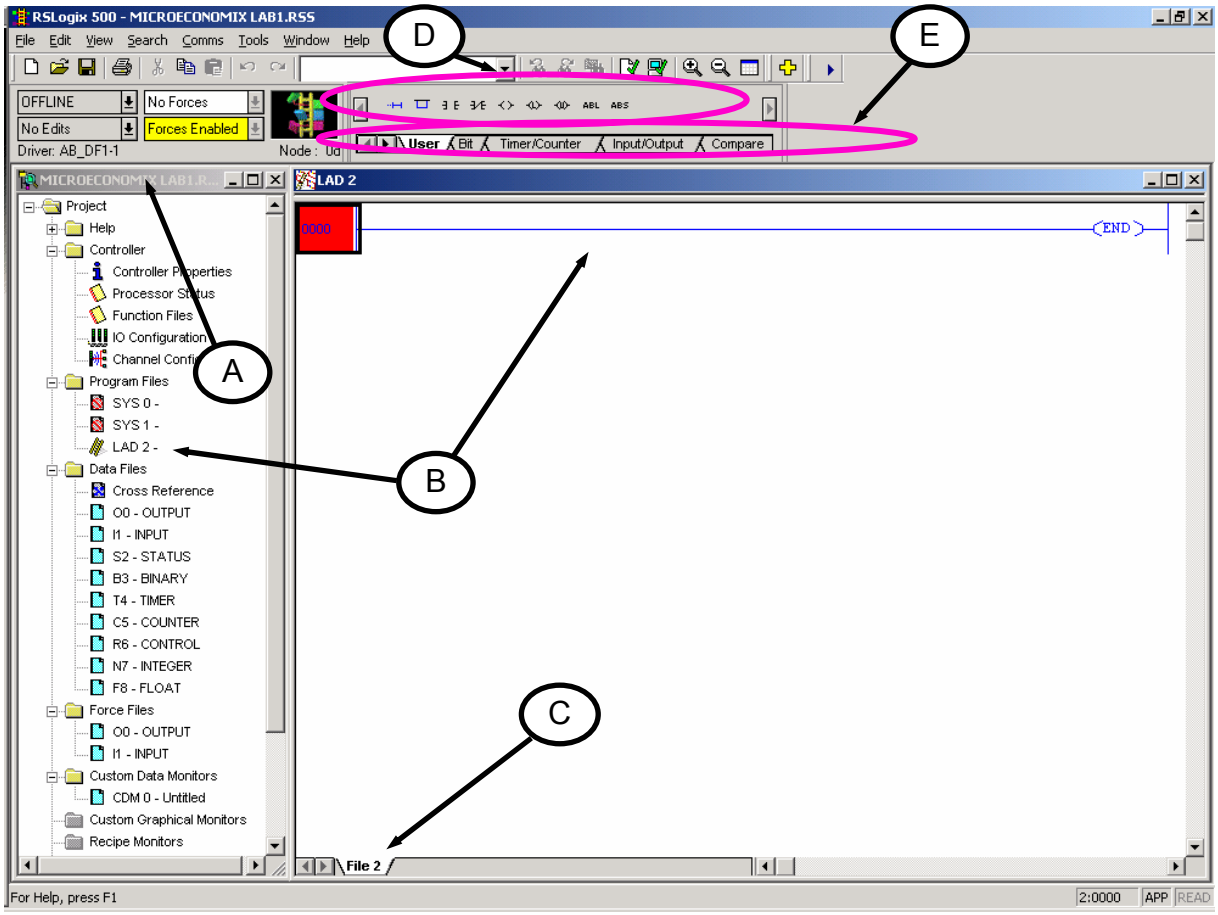
5. Jednom kada ste odabrali vaš kontroler, kliknite **OK.**
6. Iz File menija odaberite **Save As.**
7. Utipkajte '`M-vježba1`' u File name prozoru.
8. Kliknite **Save.**

File sa tim imenom već može postojati.

9. Ako File sa tim imenom već postoji, odaberite **Yes** kako bi promjenili postojeće ime.



6.2.5 Pregled vašeg novog RSLogix 500 projekta



A. Project Viewer

- **Prikazuje resurse kontrolera**
  - Controller information/setup = ciklus pregleda, informacije konfiguracije
  - Program Files = gdje se unosi ladder logika
  - Data Files = gdje se pohranjuju vrijednosti podataka
  - Force Files = gdje se može zaobići I/O (ulaz, izlaz) sa nekom vrijednošću tako da se ulaz koji je uključen vanjskim senzorom prisilno isključuje čak iako je senzor uključen. Izlaz releja također može biti prisilno uključen čak iako ga program preko kontrolera nije uključio. Gdje možete poništiti stanje pomoću "forced" vrijednosti, tako da kada je ulaz uključen pomoću vanjskog senzora, da se može isključiti čak i kada je senzor isključen. Ili se relejni izlaz može uključiti, čak iako ga program u kontroleru nije uključio.
  - Custom Data Monitors = User Configurable Data monitor registri, koji dozvoljavaju korisniku da modificira programske informacije.

B. Program Viewer

- gdje se nalaze programi
- gdje se unosi ladder logika

C. Program "TABS"

- Kada se otvori program file, kreira se "TAB".
- Osigurava brz i jednostavan pristup sadržaju program file-a.

D. Instrukcijske tipke

- Vući & Pustiti, ili dva puta kliknuti.
- Mora biti **"Program Viewer"** aktivan.

E. Tabbed Toolbar

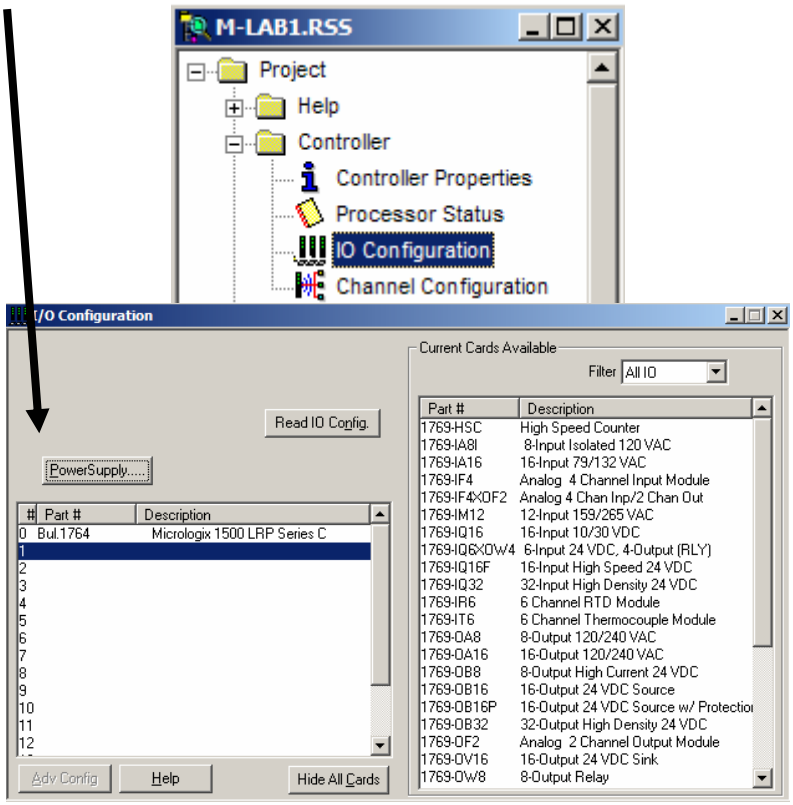
- Instrukcije grupirane po funkciji.
- Floating Toolbar Support (pomična toolbar podrška).

6.2.6 Određivanje I/O (ulazno/izlaznih) modula

Prije nego što počnete pisati vaš prvi ladder logic program ,morate naznačiti softveru sve module koji su spojeni sa kontrolerom. To možete učiniti ručno i dodati svaki modul zasebno, ali RSLogix 500 sofver nam dopušta da to izvršimo automatski "Read I/O configuration", i tako automatski konfiguriramo naš I/O (ulazi/izlaz).

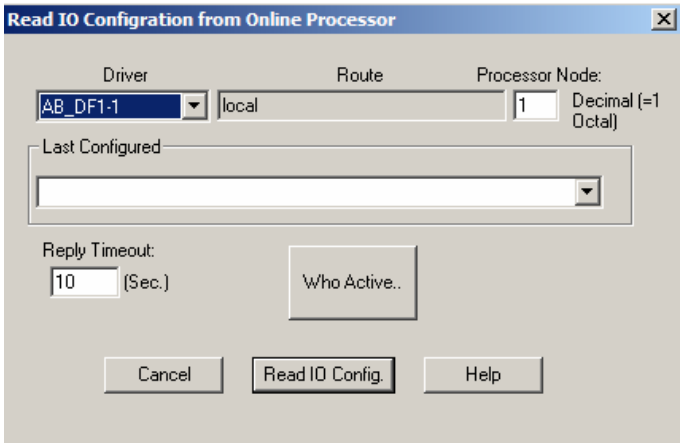
Nećemo koristiti sve dodatne I/O module u ovoj vježbi, ipak moramo pravilno konfigurirati naš mikrokontroler za hardver koji imamo.

1. Prvo dva puta kliknite na **I/O configuration** u hijerarhiji projekta, kako je prikazano dolje, da bi otvorili I/O konfiguracijski prozor.



Dodatni I/O prozor prikazan gore je za MicroLogix 1500 radnu stanicu. Ovisno o tipu kontrolera na vašoj radnoj stanici, I/O configuration(konfiguracijski)prozor će biti drugačiji ,jer različiti tipovi kontrolera koriste različite dodatne I/Omodule.

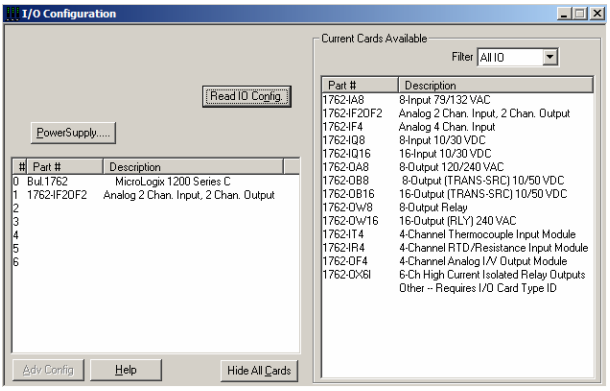
2. Iz I/O configuration menia kliknite na **Read IO Config,** kako bi konfigurirali RSLogix 500 na zahtjev I/O modula spojenih sa kontrolerom.
3. Odaberite 'AB_DF1-1' driver, kako je dolje prikazano:



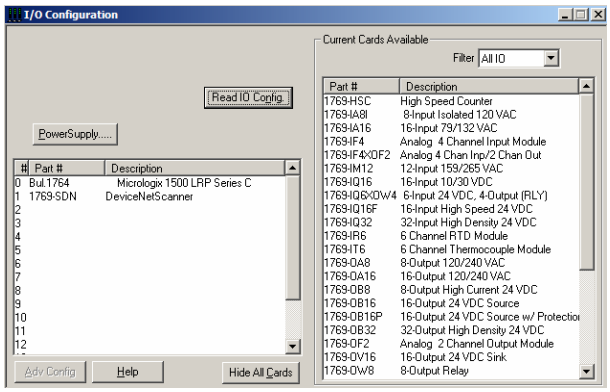
4. Kliknite na **Read IO Config** i RSLogix 500 će ponovo pronaći bili koji dodatni modul spojen sa vašim kontrolerom.

Read IO prozor će nestati i jedan od sljedećih I/O konfiguracijskih prozora će se pojaviti kako je prikazano dolje, za odgovarajući hardver koji se nalazi na vašoj radnoj stanici.

MicroLogix 1200



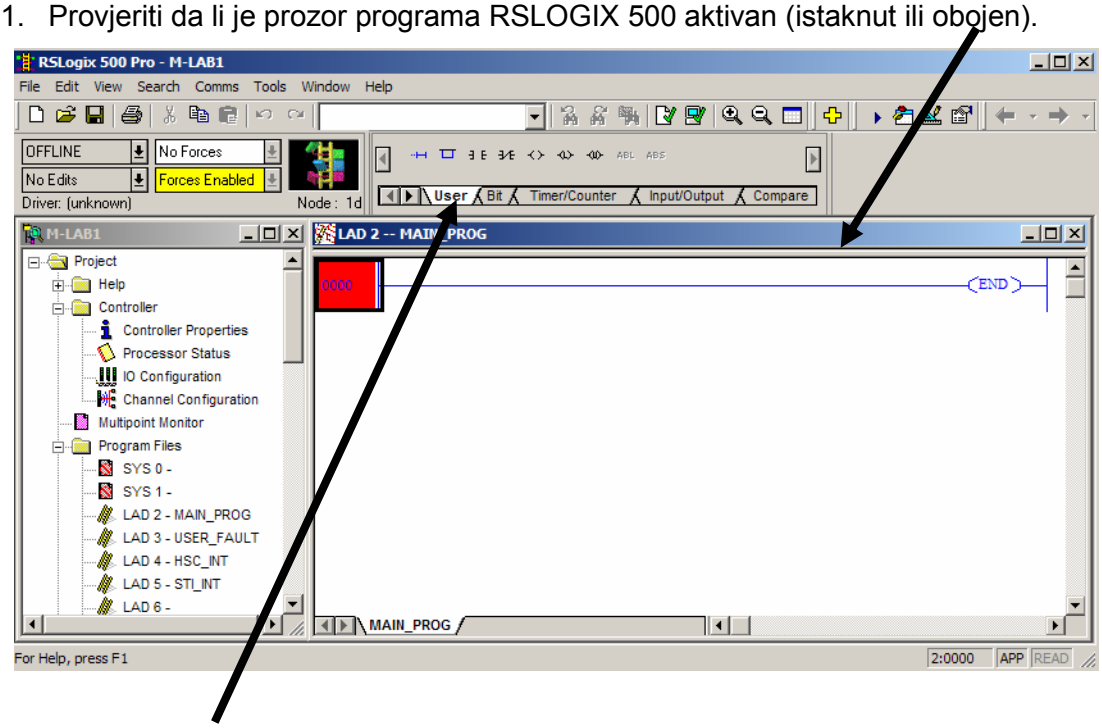
MicroLogix 1500



5. Nakon što ste pregledali I/O konfiguracijski prozor, kliknite na ‘x’ u gornjem desnom kutu prozora, kako bi ga zatvorili.

6.2.7 Kreiranje prvog logičkog kruga ladder logike

1. Provjeriti da li je prozor programa RSLOGIX 500 aktivan (istaknut ili obojen).

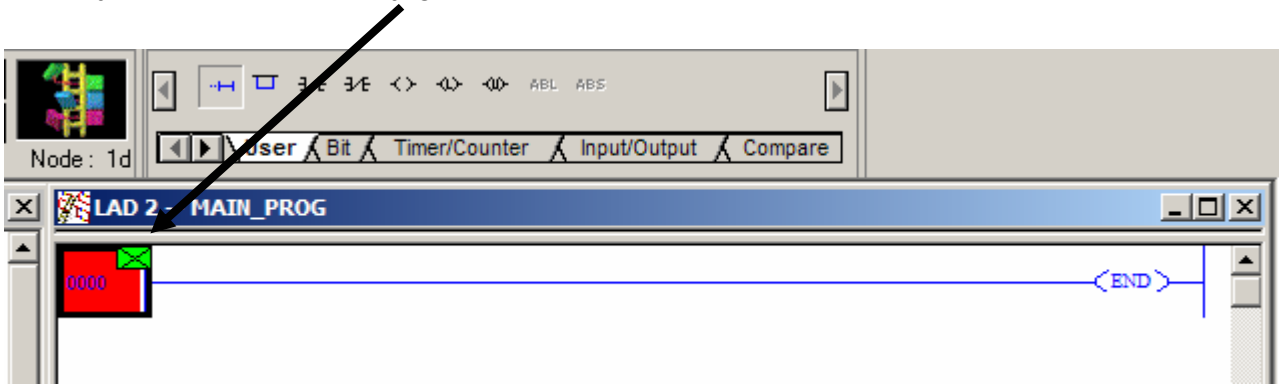


2. Kliknite na **User** tab.

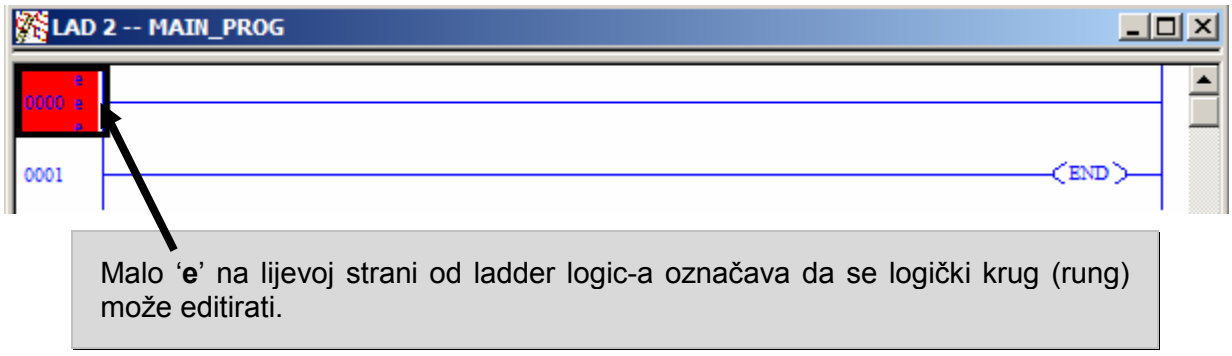
> **User** tab sadrži osnovne **ladder logic** instrukcije koje ćemo koristiti u našem programu .

3. Iz instrukcija prikazanih za **User** tab, pronađite **New RUNG** ikonu.

4. Kliknitei držite lijevu tipku miša na **New Rung,** i vucite ikonu na logičkom krugu nula (0000). Kada vidite zeleni X otpustite lijevu tipku miša. Logički krug se može smatrati kao sredstvo za dobivanje energije iz lijevog voda (engl. rail) na desni vod. U jednostavnim ladder dijagramima, žice čine upravo to.

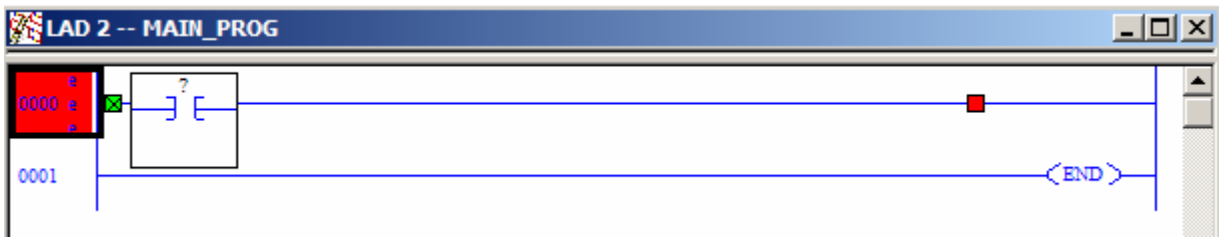


Prozor vašeg RSLogix 500 programa bi se sada trebao pojaviti, kako je prikazano dolje.

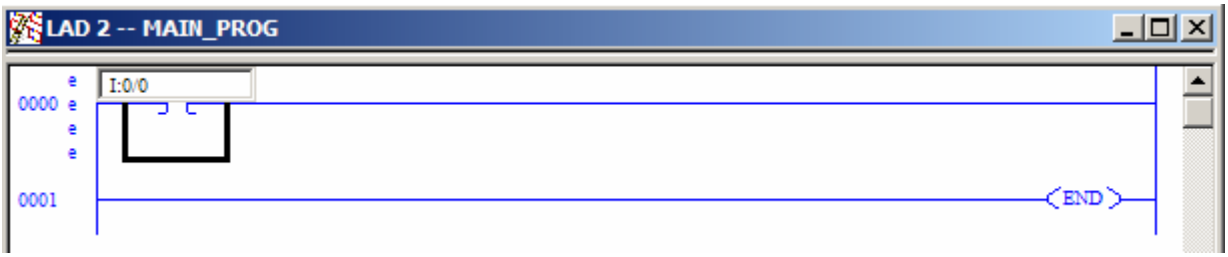


Malo '**e**' na lijevoj strani od ladder logic-a označava da se logički krug (rung) može editirati.

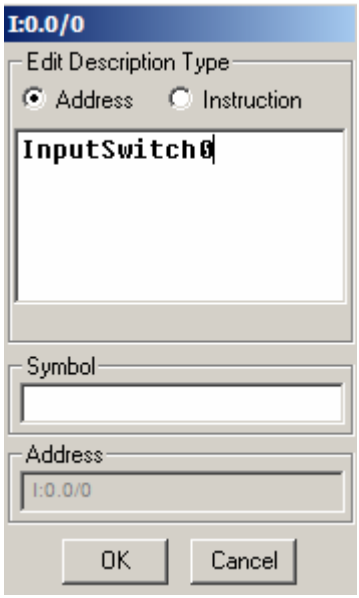
5. Naša prazana linija predstavlja kratki spoj, pa trebamo dodati uvjete, (stanja) koja definiraju kada želimo "da teče struja", da se radnja izvrši. Početi ćemo da dodamo input instrukciju na našu prazanu ladder liniju. Input instrukcija daje kontroleru informaciju koju interpretira kao "uzrok" sastavni dio od "uzroka i posljedice". Dajući spravne ulaze ("uzroke") kontroler će generirati specifične izlaze ("posljedice"). Kliknite Bit tipku da bi prikazali razinu bit instukcije. Kliknite na **XIC** instrukciju (koja izgleda ovako] [ ) u 'User" tabularu i opet vucite instrukciju do vaše nove linije. Kada vidite zeleni X, otpustite tipku miša. XIC instrukcija ispituje da li je ulaz zatvoren (poznato kao običan otvoreni kontakt).



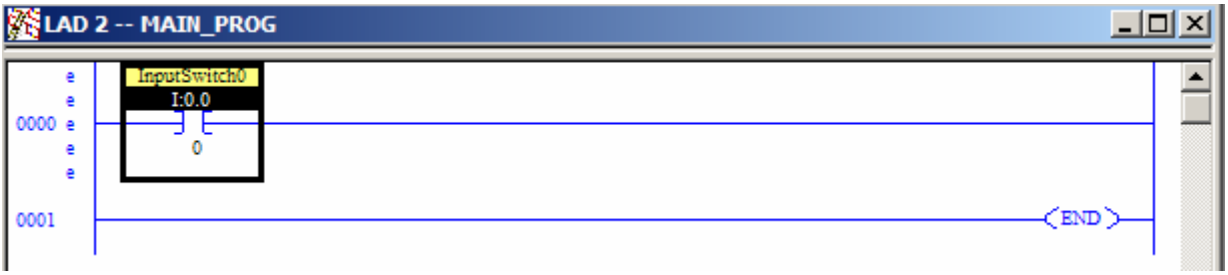
6. Sada trebamo osigurati adresu za naš novi XIC ulaz. Adresa govori ladder logici gdje da traži ulaz. PLC koristi adresiranje dok relej logika (tvrda žičana logika) koristi fizičku konekciju. Pobrinite se da je instrukcija istaknuta i utipkajte `'I:0/0'`, zatim pritisnite **Enter**.



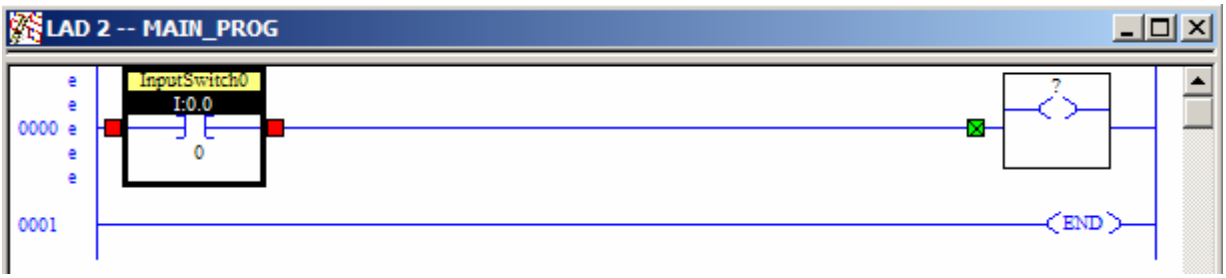
7. Sada će se od vas tražiti da unesete naziv za ovu adresu. To se ne traži u ovom zadatku, ali je dobro za vježbu imenovati vaše adrese kako biste pravilno dokumentirali svoj program, ali i zbog budućih mogućih problema. Isto tako to će vam omogućiti da kasnije "čitate" vaš program, i da ga mnogo lakše razumijete. Dakle za naš primjer utipkajmo `'InputSwitch0'i` kliknimo na **OK.**



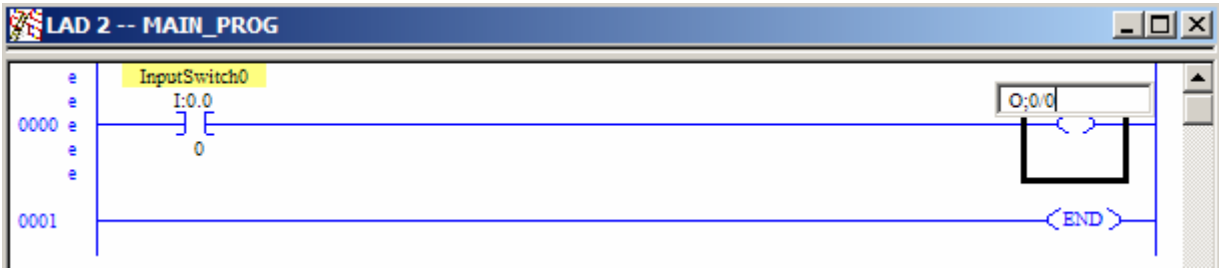
Vaš logički krug bi sada treba izgledati ovako.



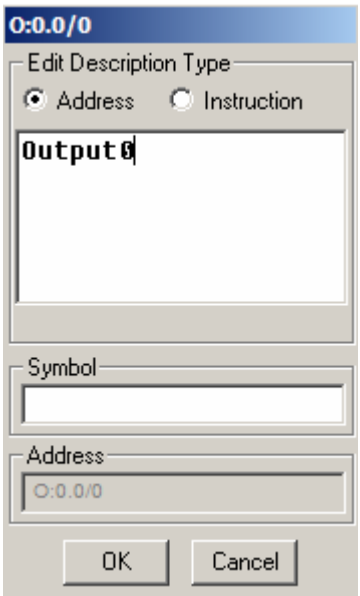
8. Sada trebamo dodati izlaz(output) na ovaj logički krug. On predstavlja "posljedicu", što smo ranije naveli. Kao što smo učinili sa XIC naredbom, iz **User BIT** tabulara. Kliknite i vucite **OTE** naredbu (Outpu Energize), koja izgleda ovako -( )-,instruction na ovaj logički krug (rung) kako je dolje prikazano. Kao i prije, kada vidite zeleni X otpustite tipku miša.



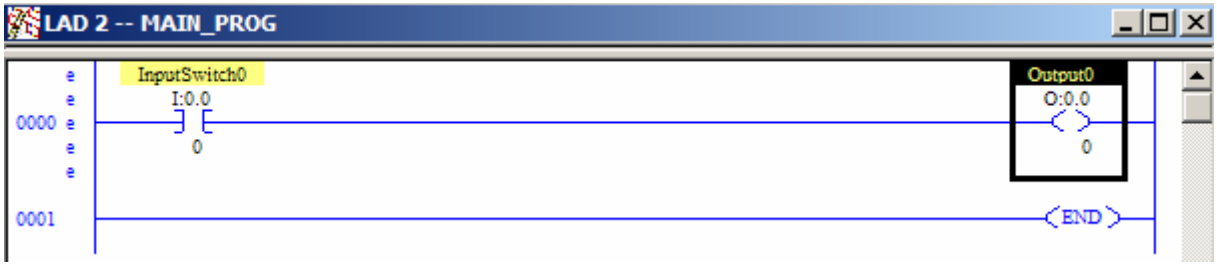
9. Ponovo trebamo osigurati adresu za naš novi OTE izlaz. Pobrinite se da je naredba aktivna i utipkajte `'O:0/0'`, a zatim pritisnite **Enter.** Napomena: Postoji razlika između slova "O" (koja se koristi za izlaz - Output), i znamenke nula "0". Oni su ovdje napisani u istom fontu u kojem će se pojaviti na vašem zaslonu.



10. Sada će se od vas tražiti da unesete naziv za ovu adresu. Ponovo, ovo nije potrebno u ovom zadatku, ali je uvijek dobro vježbati davanje imena adresama. Pa za primjer utipkajmo '`Output0`' i zatim kliknimo **OK.**



Vaša ladder linija bi sada trebala izgledati ovako. Ako vaše adrese nisu jednake onima iz primjera, pobrinite se da pravilno koristite slovo "O" i broj "0".



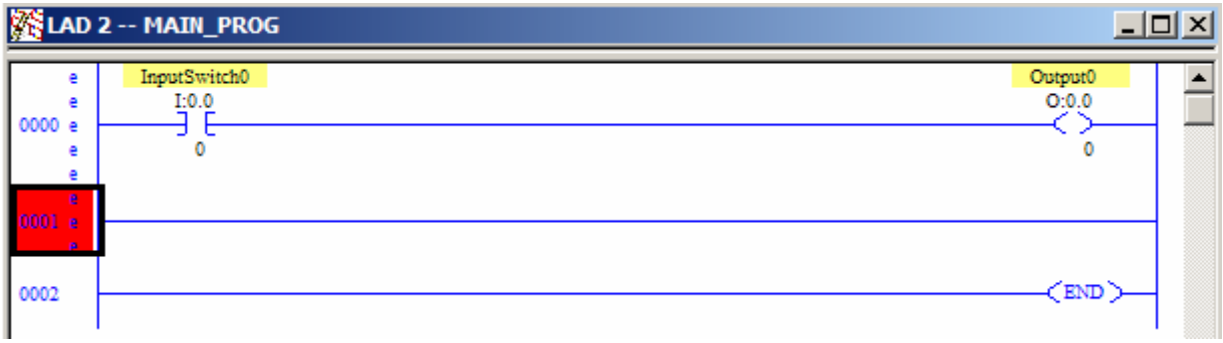
Čestitamo, upravo ste projektirali vašu prvu liniju koda ladder logike (logički krug)!

Kada se ispune uvjeti ulaza (lijeva strana kruga), izvršiti će se naredba izlaza (desna strana kruga). U ovom slučaju, OTE naredba će biti izvršena uvijek kada je XIC instrukcija ocjenjena kao "točna". Drugim riječima, kada "uzrok" (ulaz koji nazivamo Input Switch 0, na adresi I:0/0) identificira da je ulazni kontakt zatvoren, to će rezultirati kao "posljedica" (izlaz koji nazivamo Output 0, na adresi O:0/0, koji će upaliti naš izlaz.

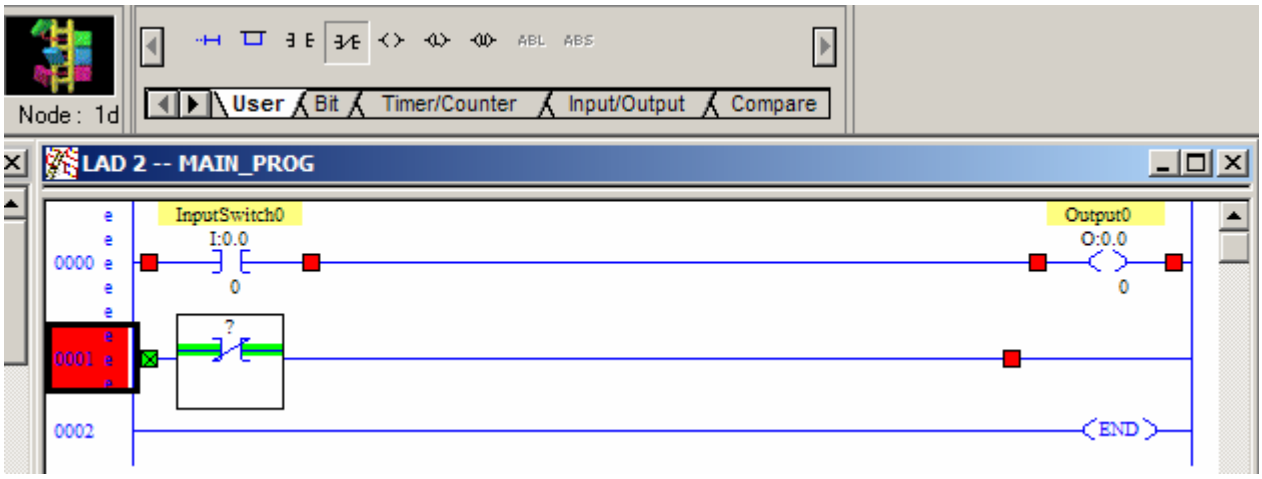
> Naš ulaz je prvi fizički ulaz na kontroleru, a izlaz je prvi fizički izlaz na kontroleru. To znamo jer adrese koje smo izabrali identificiraju ove ugrađeneu ulaze i izlaze uređaja. Naši kontroleri mogu adresirati svoje I/O (inputs/ulaze i outputs/izlaze) ponešto drugačije. Inputs (ulazi) i outputs (izlazi) u ladder logici mogu biti fizički (kao relejni izlaz prikazan gore), ili mogu biti virtualni (kao bit pohranjen u tablici podataka/data table). Možete koristiti fizički izlaz za kontrolu vanjskog uređaja. Možda ćete željete kreirati virtualne izlaze koje ćete koristiti u drugim djelovima vašeg ladder logičkog programa.

6.2.8 Kreiranje drugog logičkog kruga ladder logike

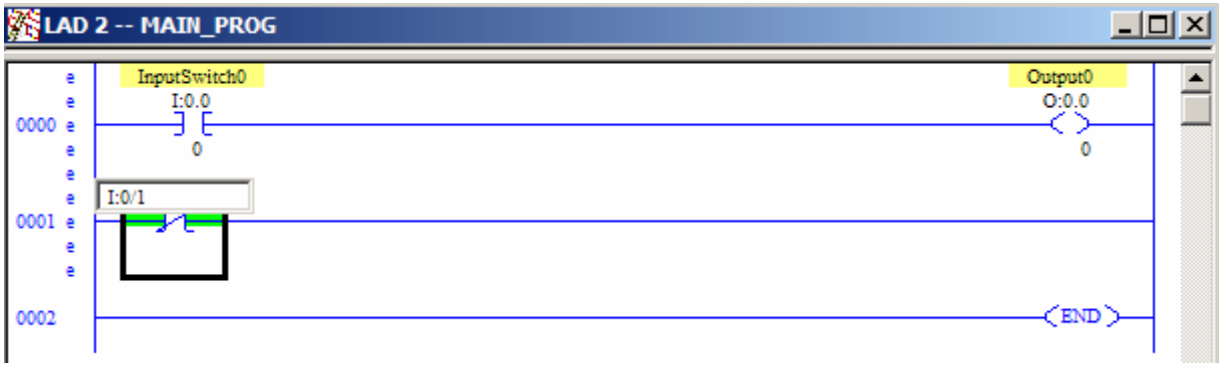
1. Trebamo pokrenuti još jedan logički krug (rung). Unesimo novi logički krug u naš program. To možete učiniti pomoću postupka koji smo opisali u prethodnom poglavlju (kliknuti/vući), ili možete pokušati drugu metodu. Odaberite rung 1 (0001) i jednostavno pritisnite INSERT tipku na vašoj tipkovnici. Vidjeti ćete prazanu liniju u vašem ladder logičnom programu, kako je prikazano dolje.



2. Dodajmo prvu ulaznu naredbu na ovaj novi logički krug (rung), kao što ste to učinili u prethodnom poglavlju. Kliknite i vucite **XIO** naredbu (koja izgleda ovako]/[ ) do linije 1 dok ne vidite zeleni X tada otpustite tipku miša. XIO naredba provjerava da li je ulaz otvoren (poznato kao običan zatvoren kontakt).

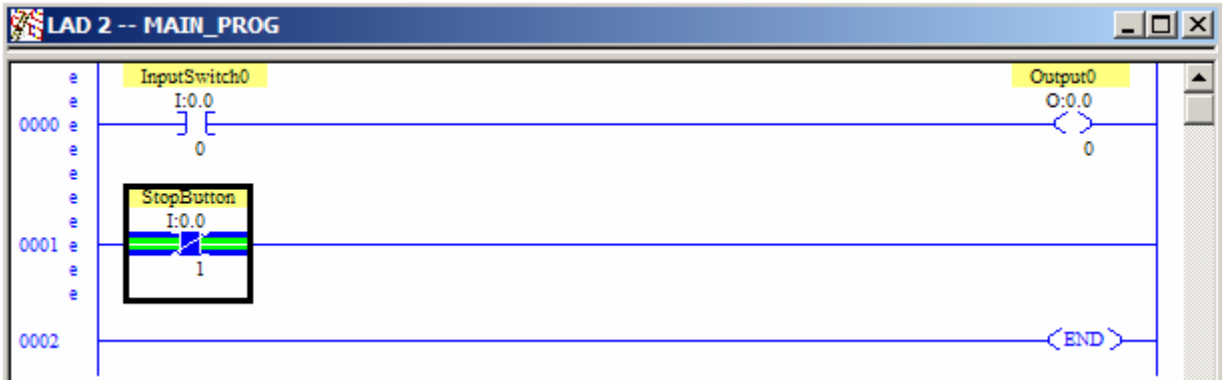


3. Za nedavno dodanu i aktivnu XIO naredbu, upišite adresu: `'I:0/1'` i pritisnite **Enter.**

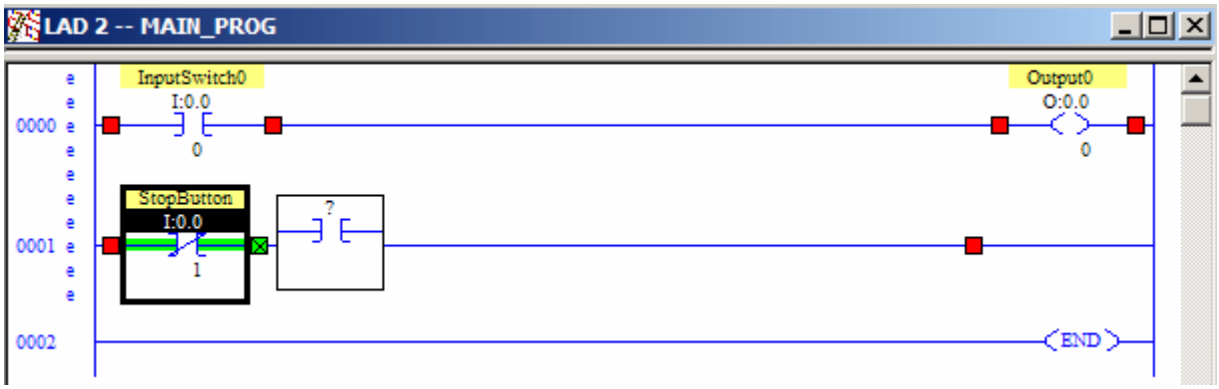


4. Tražiti će se od vas da imenujete ovaj novi ulaz. Utipkati ćemo `'StopButton'`, zatim kliknuti **OK.**

Vaš logički dijagram bi trebao izgledati ovako:

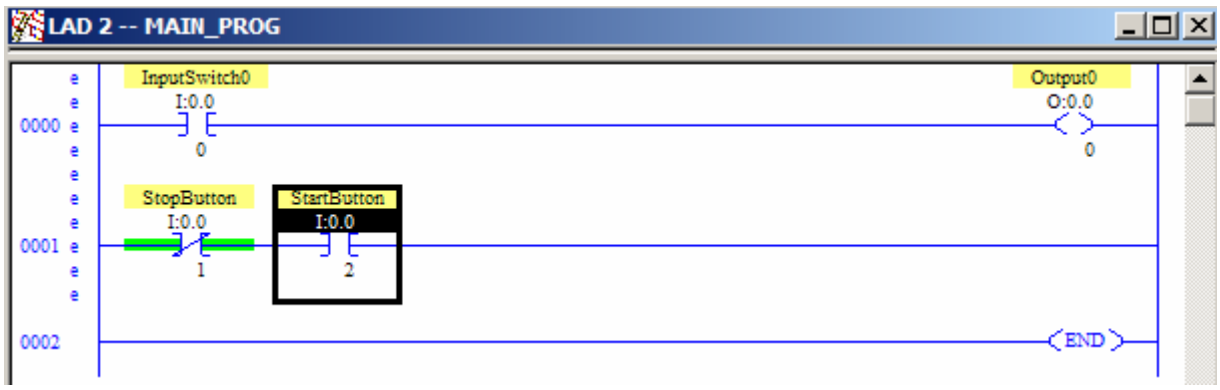


5. Dodajmo drugu ulaznu naredbu desno od **XIO** na rung 1. Kliknite i vucite **XIC** naredbu (provjeri da li je zatvoreno) dolje do rung-a 1, desno od XIO naredbu, sve dok ne vidimo zeleni X, tada otpustite tipku miša.

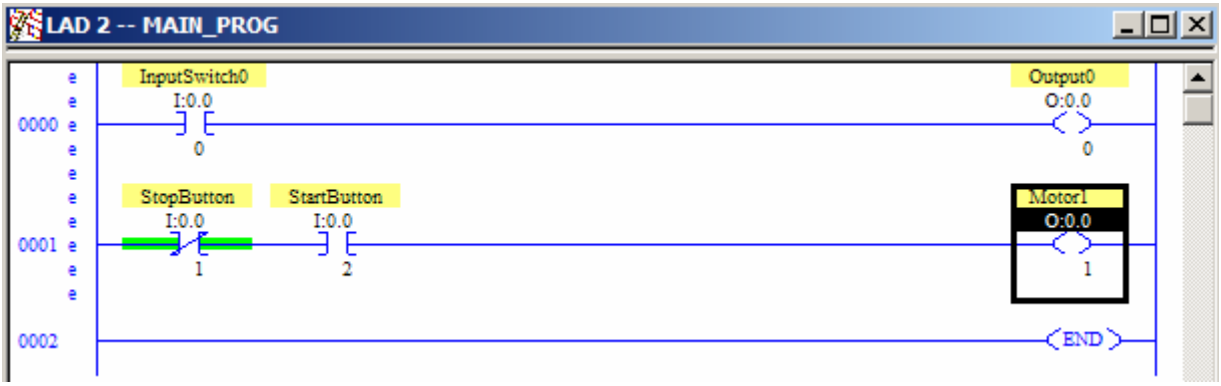


6. Koristeći ono što ste naučili iz prethodnih koraka, adresa ove naredbe neka bude `'I:0/2'`, zatim kliknite `'StartButton'` (start tipkalo). Kada završite, vaš bi logički krug (rung) trebao izgledati ovako.

> Stavljanje serijskih uvjeta na logički krug omogućuje nam da uočimo "AND" uvjet. U ovom slučaju tražimo da nije Stop Button (tipkalo) uključeno, I (AND) da je Start Button (tipkalo) uključeno. Oba uvjeta moraju biti "istinita" da bi logički krug bio "istinit".



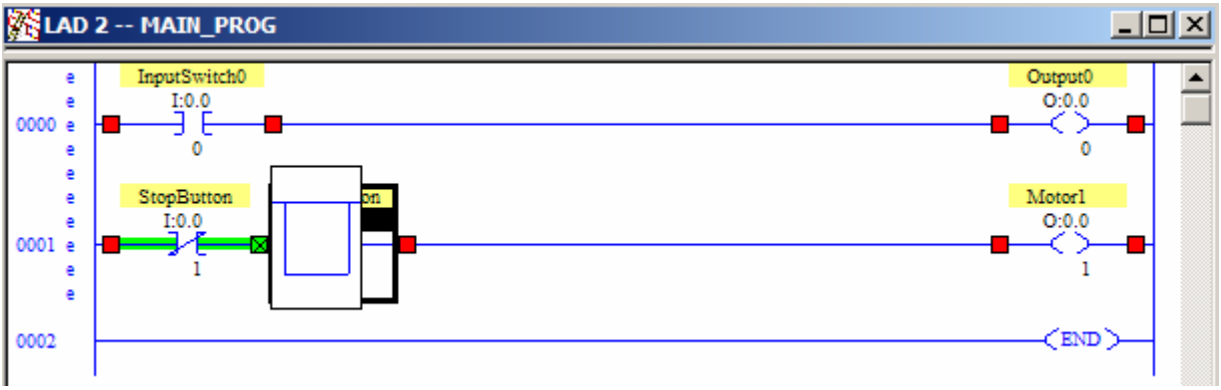
7. Dodajte **OTE** naredbu (Output Energize) na logički krug 1.
8. Unesite sljedeću adresu: `'O:0/1'`, naziva: `'Motor1'`, kada završite, vaš logički krug bi trebao izgledati ovako:



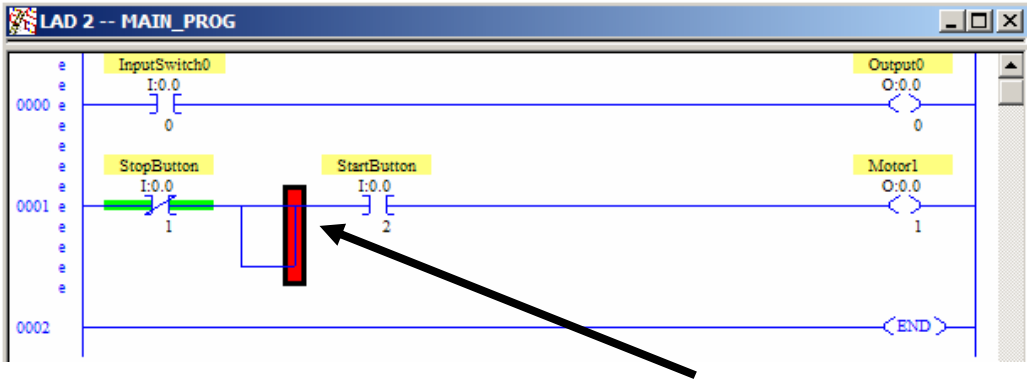
Sada ćemo dodati granu logičkom krugu 1.

> Grane omogućuju "ILI/OR" programiranje. Na primjer, koristiti ćemo granane logičke krugove ako želimo ukazati da se *bilo* Push Button 1 *ili* Push Button 2 mogu koristiti za uključenje izlaza.

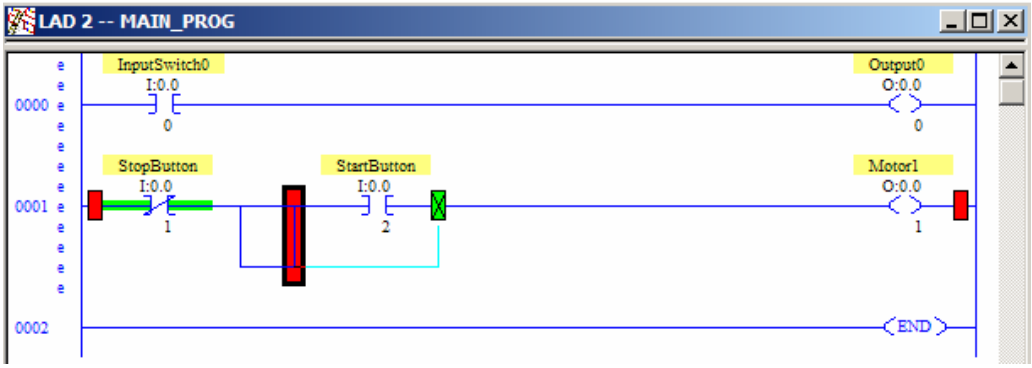
9. Kliknite na User tab (na lijevo od Bit tabulara). Zatim kliknite i držite lijevu tipku miša i vucite **Rung Branch**  tipku između XIO i XIC naredbi na logički krug 1. Kada vidite zeleni X, otpustite tipku miša.



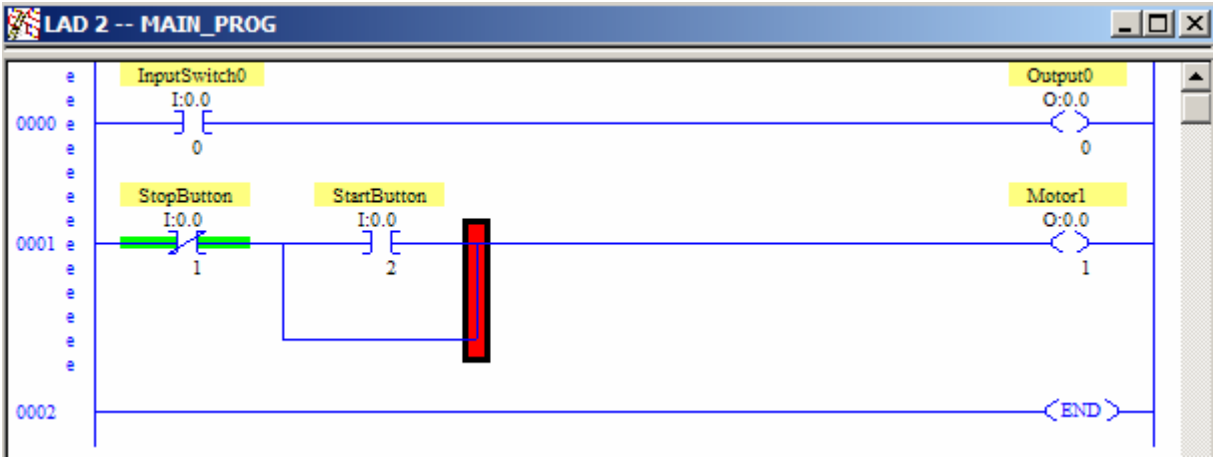
Vaš logički krug bi trebao izgledati ovako:



10. Sada moramo identificirati da li želimo koristiti "oba/ili" naredbu. Da bi to učinili trebamo pomaknuti granu oko Start Button-a (start tipkala). Kliknite i držite desnu stranu grane. Vucite taj dio grane na desno od Start Button-a. Kada vidite zeleni X, otpustite tipku miša.



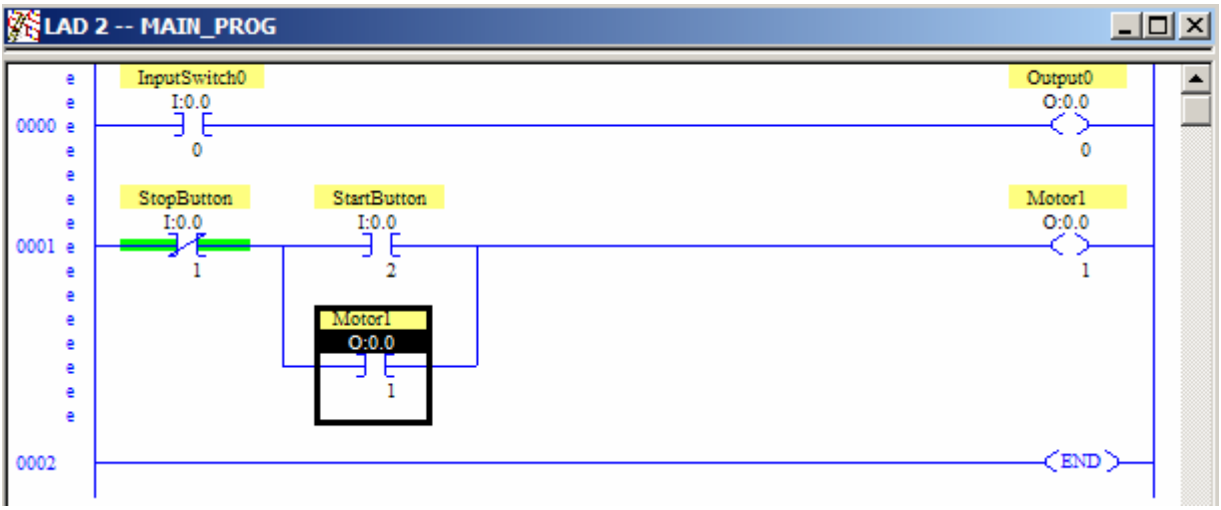
Vaš logički krug bi trebao izgledati ovako:



11. Sada trebamo dodati ulaznu naredbu na našu granu. Koristeći ono što ste naučili iz prethodnih koraka. Kliknite i vucite **XIC** naredbu na novu granu koju ste upravo kreirali.

12. Adresirajte XIC instrukciju kao: `'O:0/1'`.

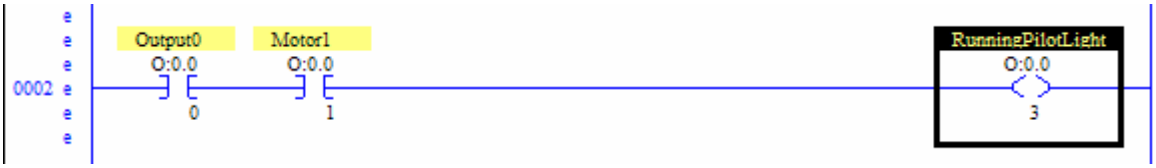
Logički krug bi trebao izgledati ovako:



Primjetiti ćete da ste unjeli istu adresu za XIC naredbe kao i za izlaznu adresu OTE naredbe. To je sposobnost PLC-a. Ono što ste upravo ostvarili je je ekvivalentno sa pomoćnim kontaktom kod startanja motora koji omogućava projektiranje zatvorenog (engl. seal in) strujnog kruga. Dodajući ovu granu na OTE adresu, motor će ostati upaljen kada se start button (tipkalo) otpusti, sve dok se stop button (tipkalo) ne pritisne.

6.2.9 Kreiranje trećeg logičog kruga ladder logike

1. Koristeći ono što ste do sada naučili dodajte još jedan logički krug kako je prikazano dolje.

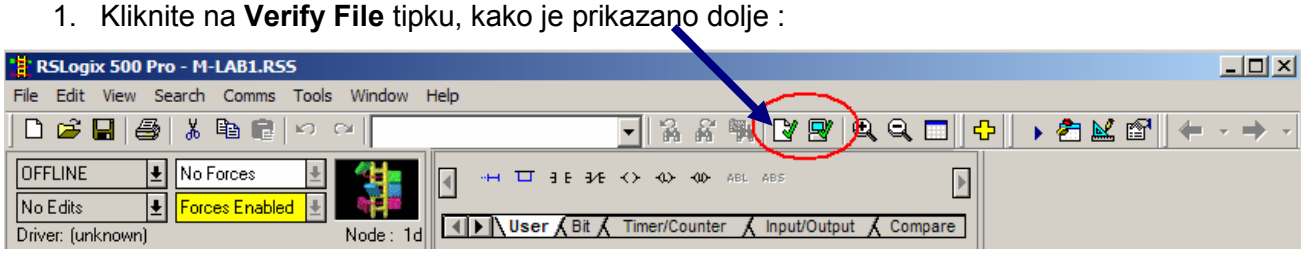


6.2.10 Provjeravanje vašeg ladder logic programa

Verifikacija ili valjanost programa, provjerava program koji ste napisali tražeći pogreške. Nakon što je provjera završena otvoriti će se prozor s “rezultatima” koji vam daje informacije o pogreškama ili propustima koji su nađeni kada je vaš program bio ispitan pomoću softvera.

Postoje dvije metode verifikacije (provjere) programa. Prva metoda:provjerava file u kojem trenutno radite i isključivo taj file. Druga metoda provjerava sve file-ove (Main i Subroutines) za projekt koji ste kreirali. Zato što se naš program jedino nalazi (pripada) u File #2, pa ćemo koristiti prvu metodu provjere programa.

1. Kliknite na **Verify File** tipku, kako je prikazano dolje :

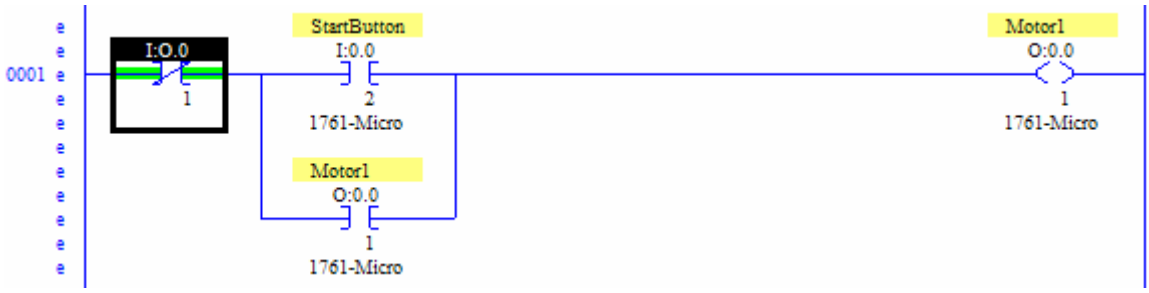


Kada je provjera završena i ako nije bilo nikakvih pogrešaka svi edit markeri u programu (‘e’ na lijevo od ladder linija), pojavit će se na dnu RSLogix 500 softver ekrana.

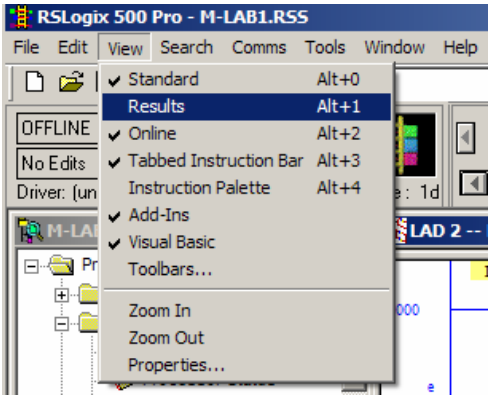
Verify has completed, no errors found

Ako nije identificirana niti jedna pogreška u vašem logičkom programu, idite na idući dio: “pohrana vašeg rada”.

Ako imate pogreške u vašem logičkom krugu, oni logički krugovi koji sadrže pogrešku još će uvijek pokazivati edit marke (mali "e" pokraj broja logičkog kruga) kako je prikazano dolje.



2. Otvorite prozor s 'Rezultatima' iz menija View. **View→Results**, ili jednostavno pritisnite: **ALT+1.** Ovaj prozor će se otvoriti automatski ako su nađene pogreške u programu.



RSLogix će vas direktno odvesti na pogreške u vašem programu. Jednostavno kliknite na:error message u 'Verify Results' prozoru, i primjetiti ćete da je RSLogix 500 istaknuo naredbu koja sadrži pogrešku.

3. Vratite se na prethodnu vježbu i ispravite vaše pogrešku(e).

Opaska: Uobičajna pogreška je korištenje slova umjesto brojeva, kao na primjer: O umjesto 0 (nula).

4. Nakon što ste ispravili vaše pogreške provjerite iznova vaš file. Jednom kada su sve pogreške ispravljene, krenite na idući dio:"Pohrana (engl. Saving) vašeg rada (programa)".

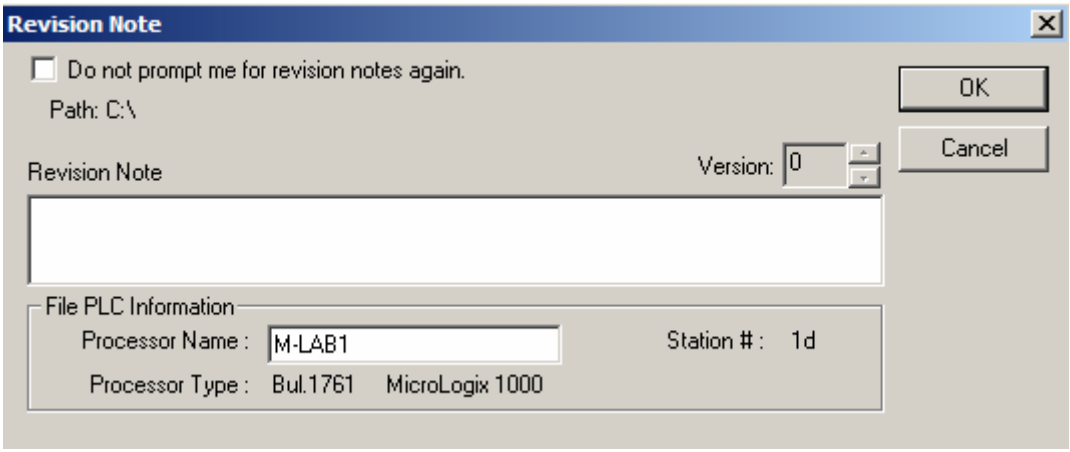
6.2.11 Pohrana vašeg programa

Vaš projekt još nije pohranjen na tvrdom disku računala. Pohranite (snimite) vaš program kako bi spriječili da se vaš rad izgubi.

1. Kliknite **Save** tipku  ili iz glavnog menija odaberite **File → Save.**

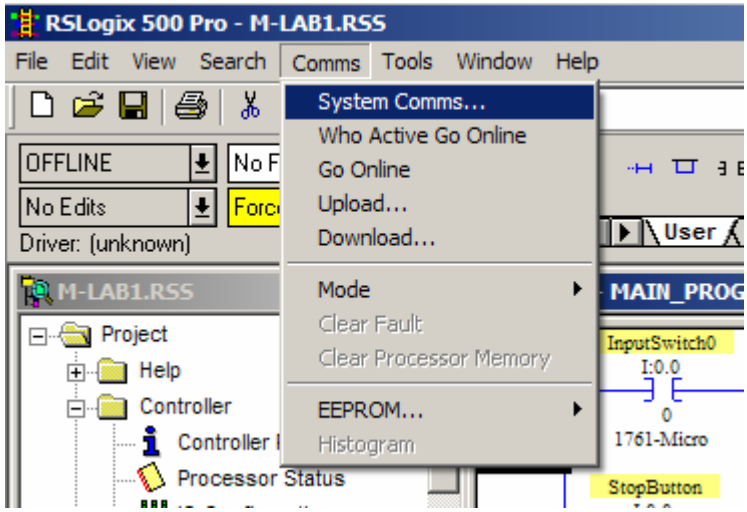
Revision note prozor će se otvoriti. RSLogix 500 softver je automatski konfiguriran tako da vam dopušta da unesete bilješke (engl. enter notes) o izmjenama vašeg programa, i da dođete do back-upa stare verzije vašeg programa dok vi unosite promjene. To vam omogućuje da se brzo vratite na prethodnu verziju vašeg programa, ako učinite pogrešku. Broj revision kopija je moguće konfigurirati, i može se isključiti ako ne želite imati ovu funkciju aktivnu.

2. Kliknite **OK** u Revision Note prozoru.



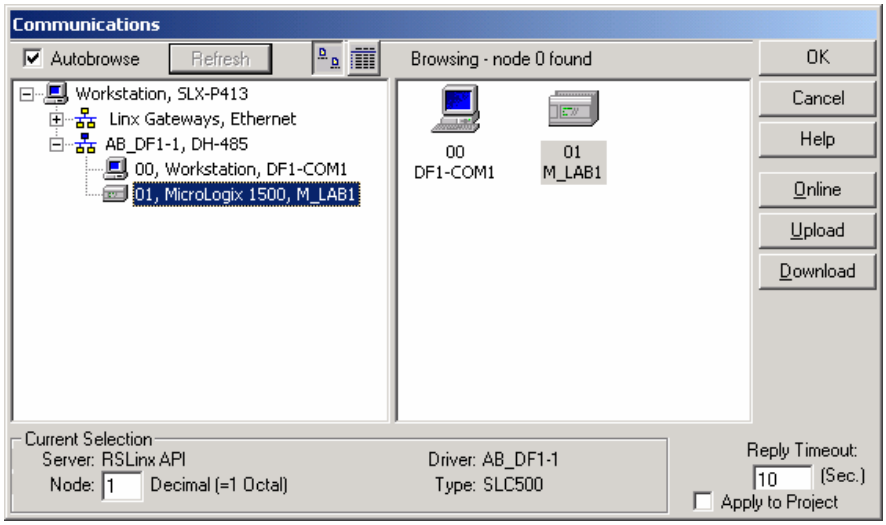
6.2.12 Download vašeg ladder logic programa na PLC

1. Iz RSLogix 500 glavnog menija odaberite. **Comms → System Comms.**



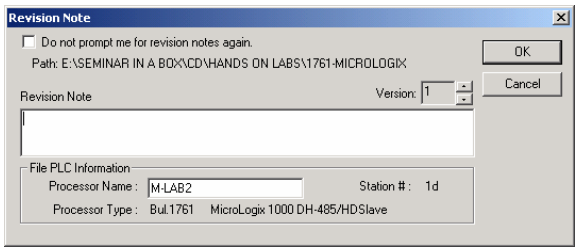
2. Ako **AB_DF1,DH485** driver nije otvoren, proširite driver tako da kliknete na '**+**' predznak ispred drivera.

3. Istaknite PLC kod Node **01**.

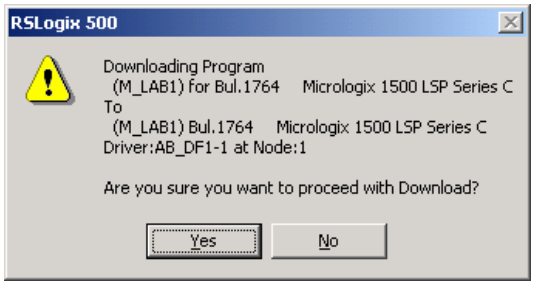


Ono što ćete vidjeti prikazano na vašem računalo je ime file-a koji se trenutno izvršava na vašem kontroleru, a ne nužno ovo što je prikazano gore. Nakon što "skinemo"(download) naš program, ime kontrolera će se promijeniti u ono u što smo ga programirali u prethodnim koracima.

4. Kliknite **Download.**
5. Kliknite **OK** u Revision Note prozoru.

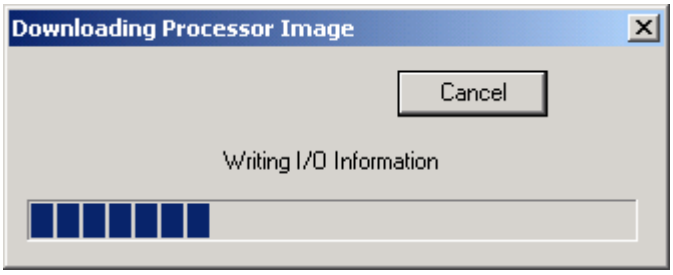


6. Kliknite **Yes,** kako bi potvrdili da želite presnimiti (download-ti) vaš program preko postojećeg programa u vašem PLC-u. Ovaj prozor će se pojaviti svaki puta kada se program snima (download-ira) na PLC.



Ako je PLC u RUN modu prije download-a, RSLogix 500 će tražiti da se promijeni u Program mod. MicroLogix 1200 kontroler nema Mode sklopku, pa je softver jedini način da se promijeni kontrolerov operacijski mod, iz Run u Program. Micrologix 1500 kontroler ima mode switch, i kada je u Remote Run/Prog modu software je u mogućnosti promijeniti mod kontrolera. Ako je Mode sklopka na MicroLogix 1500 kontroleru u Run poziciji ili u Program poziciji, softverski program neće biti u mogućnosti promijeniti operacijski mod.

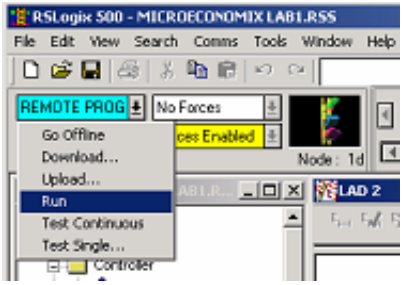
7. Kliknite **Yes,** ako vas se traži da promijenite mod.
8. Download Verification Progress Bar će se pojaviti dok se događa download.



9. Ako i kada se pojavi prozor **'Apply channel configuration to online processor'**, odaberite **'Don't Apply'.**
10. Kliknite **No,** kada vas se traži da promijenite u run mod. Mi ćemo ručno promijeniti operacijski mod (engl. operating mode) u sljedećoj vježbi.
11. Odaberite **Yes,** kada RSLogix 500 traži dopuštenje i da ide online.

6.2.13 Mijenjanje PLC-a iz programa u izvršni mod

1. Kliknite na strelicu – koja pokazuje prema dolje, pokraj riječi **'REMOTE PROG'**.



Primjetite da ima (3) Run odabira.

“Run” PLC ‘scans’ (skenira, pomno ispituje) program i izlazi su uključeni (engl. enabled).

“Test Continuous” PLC ‘scans’ program i izlazi su isključeni (engl. disabled).

“Test Single Scan” PLC izvršava jedan ciklus skeniranja, sa isključenim izlazima.

2. Kliknite **Run.**
3. Bit ćete upitan: “Da li ste sigurni da želite promijeniti procesor mod u RUN?-kliknite **YES.**

6.2.13.1 Praćenje i testiranje vašeg logičkog kruga

Sa PLC –om u ‘Remote Run’, možete pratiti ili editirati podatke pomoću kontrolera. Ovo vam omogućuje da učinite:program debugging changes i da promijenite varijable podataka dok se nalazite u run modu.

Kada se pojavi “green bars” na bilo kojoj strani od elementa logičkog dijagrama, to ukazuje na logički kontinuitet, a to pomaže odrediti kako određena aplikacija radi, te je projektirano da vam pomogne u debagiranju vaše aplikacije logičkog dijagrama. U relej logičkom modelu, to je isto kao "protok struje".

Sada ispitajmo logički kontinuitet u naredbama! (engl. ladder logic in operation).

RUNG 0/LOGIČKI KRUG 0

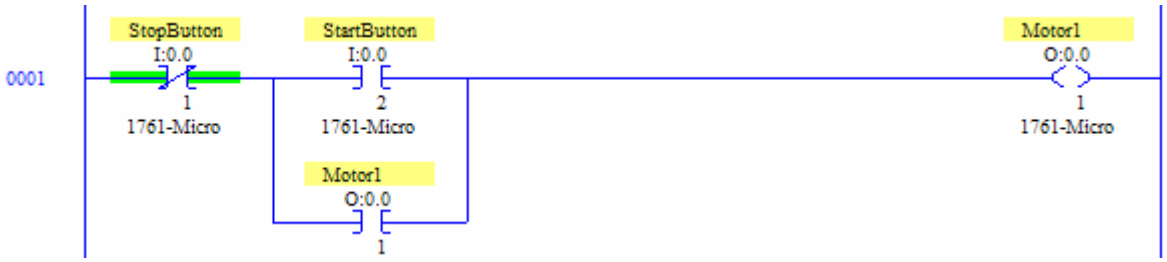
Uključite input switch #0 na input simulatoru, i ispitajte prednju stranu vašeg MicroLogix kontrolera. Output #0 će se uključiti, i na vašem kontaktnom dijagramu vidjeti ćete aktivne ‘InputSwitch0’ i ‘Output0’, što ukazuje da su input i output switch bili uključeni. Ako isključite ‘Inputswitch0’, tada će se ‘Output0’ isključiti.

RUNG 1/LOGIČKI KRUG 1

Pažljivo promatrajući logički kontinuitet, **Toggle** (ON zatim OFF) ‘StartButton’ (start tipkalo) na trećem prekidaču input simulatora. Primjetite da su se ‘StartButton' (start tipkalo) i ‘Motor1’ na MicroLogix kontroleru uključili.

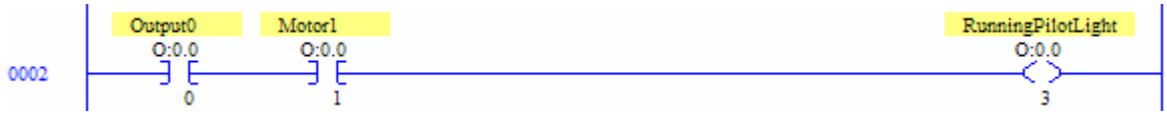
Opaska: Inputs (ulazi) i Outputs (izlazi) su numerirani 0,1,2, itd. To znači da je adresa I:0/2 zapravo treća **toggle switch** na input (ulaznom) simulatoru.

Čak I nakon što ste isključili ‘StartButton’ (start tipkalo) motor ostaje uključen. Logička odluka između StartButton-a i Motor1 u našem strujnom krugu je "ILI". Ako je StartButton uključen ili je Motor1 uključen, motor bi trebao raditi. Ovo je pravi primjer 'latching circuit', također nazvanog kontrolni krug sa 3-žice. Ako prekinete ‘StopButton’ (stop tipkalo), (drugi prekidač) na input (ulaznom) simulatoru, vidjeti ćete da se ‘Motor1’isključio.



RUNG 2/LOGIČKI KRUG 2

Dva ulaza u ovom logičkom krugu su izlazi logičkog kruga 0 i 1. 'Running/aktivan PilotLight' će se uključiti kada su dva ulaza istinita. Dakle drugim riječima, izlazi logičkog kruga 0 i logičkog kruga1 moraju biti uključeni (ON) ili istiniti (TRUE) da bi upalili naš 'RunningPilotLight'. Ovo je primjer "AND" logičke logičkog kruga.



6.3 Vježba 3: Timer, brojač (counter) i limitiranje ladder logike

Sada ćemo uzeti vaš program iz vježbe #2, modificirati dva logička kruga, i projektirati neku novu logičku naredbu. Također ćemo programirati timer, i counter (brojač).

Zatim ćemo to iskoristiti u našem programu da bi vidjeli kako se one mogu koristiti da se omoguće jedinstvene kontrolne sposobnostil. Posljednje što ćemo učiniti je programirati **Limit command**. Limit command je jedna od nekolicine moćnih instrukcija za uspoređivanje dostupnih kod ovih MicroLogix kontrolera.

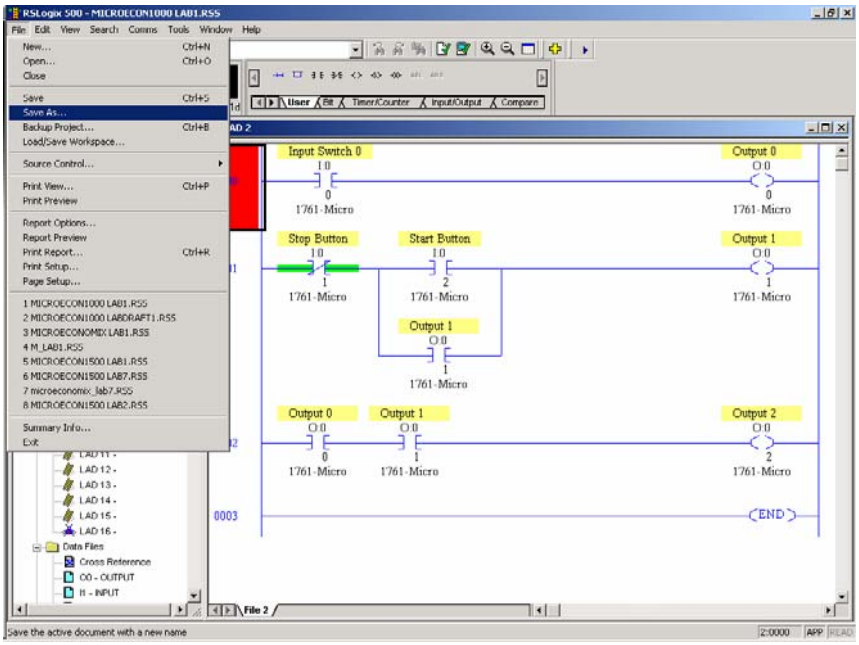
6.3.1 S PLC-om prijeđite na off-line mod rada

Ako ste ONLINE sa MicroLogix morate biti OFFLINE, kako bi dovršili ovaj postupak.

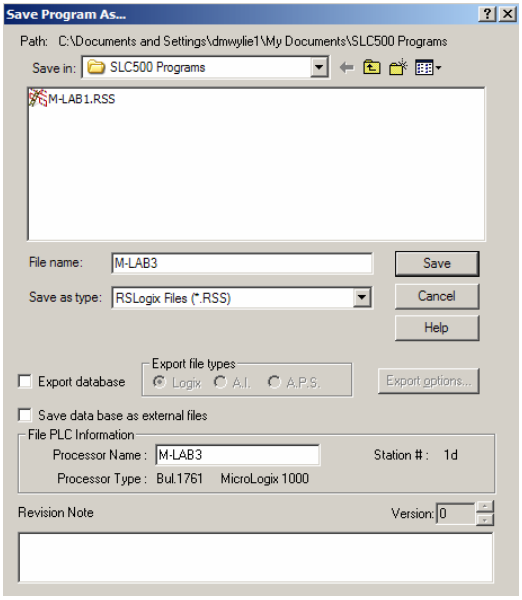
1. Kliknite na stelicu koja pokazuje prema dolje, a nalazi se pokraj zelenog prozora u kojojem piše **REMOTE RUN.**
2. Odaberite **Go Offline**.
3. Ako vas se traži da "Save Changes" (pohranite preinake), kliknite **No.**

6.3.2 Kopiranje i preimenovanje vašeg programa ladder logike

1. Otvorite **File** meni.
2. Odaberite **Save As**, kako bi kreirali kopiju.



3. Upišite novi naziv ‘M-LAB3’ u File name prozor, kako je prikazano dolje.
4. Upišite ‘M-LAB3’ u Processor Name prozor, kako je prikazano dolje.



5. Kliknite **Save.**

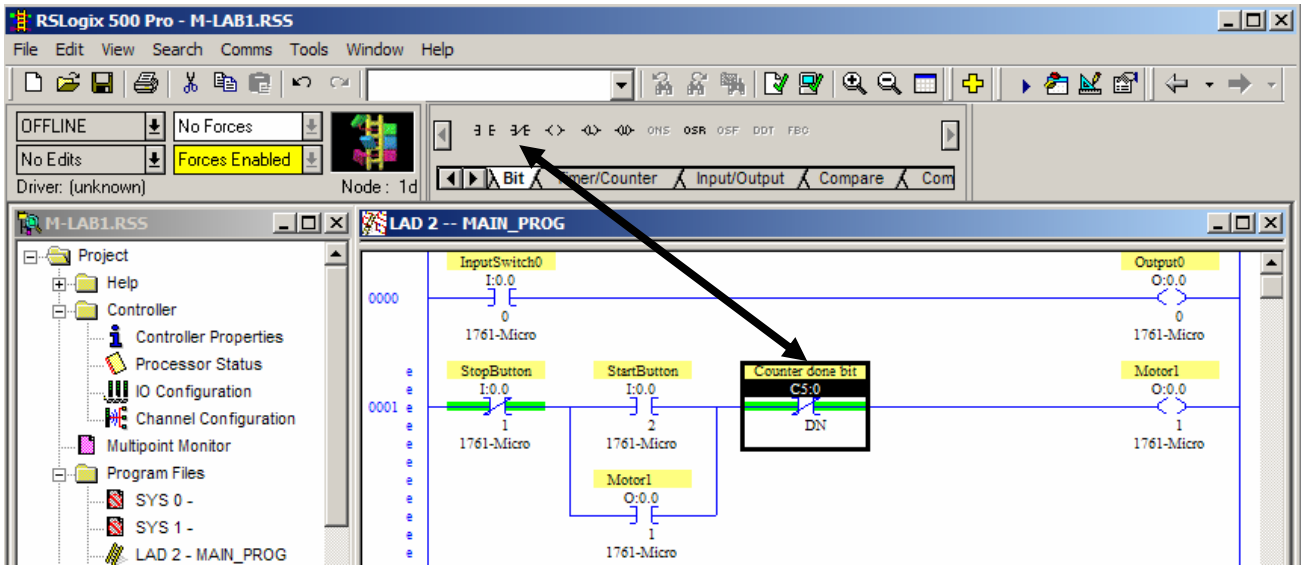
6.3.3 Modificiranje postojećeg programa ladder logike

Prvo dodajmo XIO naredbu logičkom krugu1.

Pobrinite se da je prozor u kojem se nalazi program (Program Window) aktivan. Ako nije, kliknite na Title bar kako bi odabrali ovaj prozor.

1. Kliknite na **User** tipku.
2. Kliknite, držite lijevu tipku miša i vucite **XIO** tipku imeđu grane i OTE. Upamtite, kada vidite zeleni prozor, otpustite tipku miša.

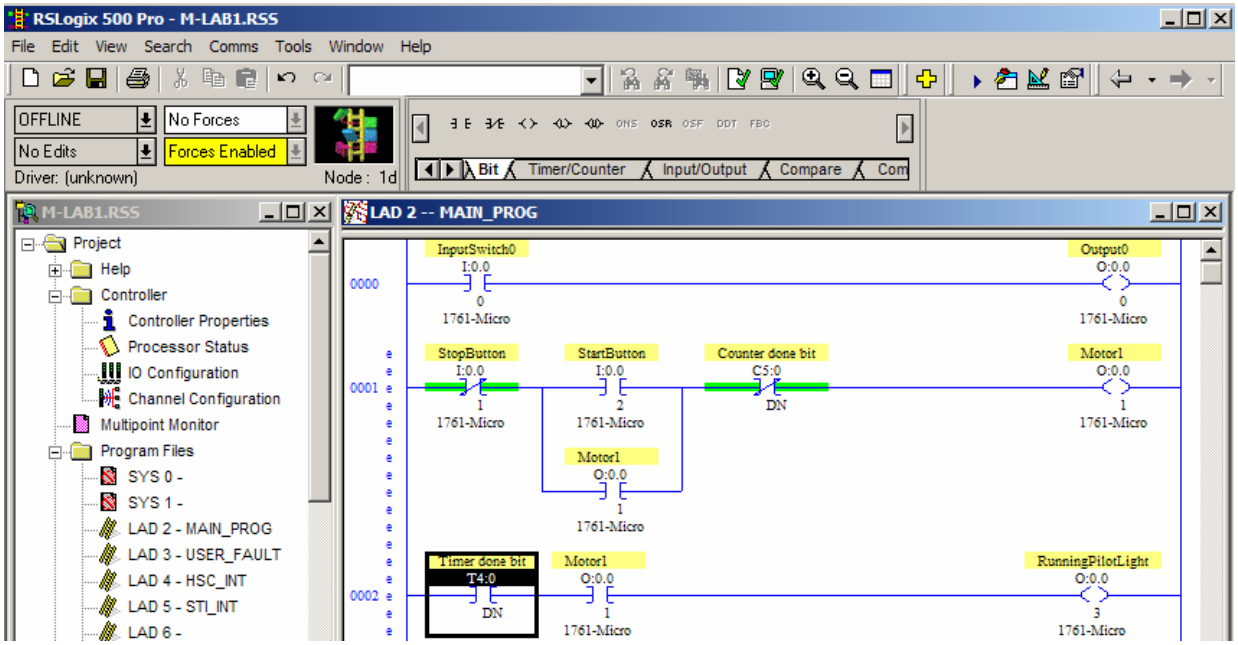
3. Dok je instrukcija aktivna, upišite: ‘C5:0/DN’, zatim **Enter**. Ovo je adresa XIO (provjeri da li otvoreno) naredbe.



4. Kada vas se traži da da unesete naziv, unesite: ‘Counter done bit’ kako je prikazano i pritisnite **OK**.

Modificirajmo/izmijenimo prvu XIC naredbu u logičkom krugu 2.

5. Dva puta kliknite na **XIC** naredbu, tako ćete otvoriti text edit prozor naredbe.
6. Upišite: ‘T4:0/DN’, zatim **Enter**. To će overwrite/prebrisati postojeću adresu. Ovo je jedna od bitnih značajka mikrokontrolera, mogućnost da se promijeni operacija programa kroz softver. Da bi učinio isto u relej logici, električar bi trebao isključiti/diskonektirati žicu iz otvorenog kruga (ili Inpu Switch 0 iz logičkog kruga 0) i pričvrstiti žicom izlaz timera. Mikrokontroler ima prednost u tome što mu je timer već ugrađen.



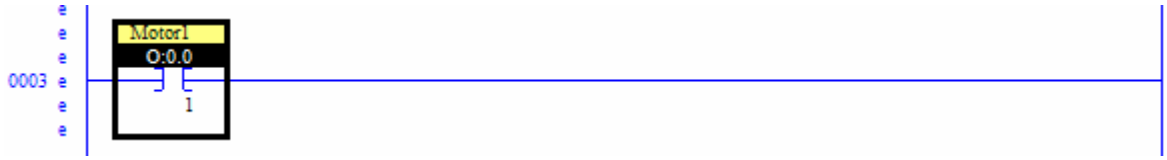
7. Kada vas se traži da unesete naziv, unesite: `'Timer done bit'` kako je prikazano dolje, i pritisnite **OK.**

6.3.4 Dodavanje timera vašem programu ladder logike

1. Kliknite, držite i vucite **New Rung (novi logički krug)** tipku preko “0003” kruga.
2. Kliknite, držite i vucite **XIC** tipku na lijevu stranu logičkog kruga koji ste upravo kreirali.
3. Dok je naredba aktivna, upišite `'O:0/1'`, zatim **Enter**.

> Primjetite, čim unesete adresu ‘O:0/1’ naziv ‘Motor1” se automatski pojavio. Jednom kada adresa ima naziv , taj naziv nosi kroz cijeli program ladder logike.

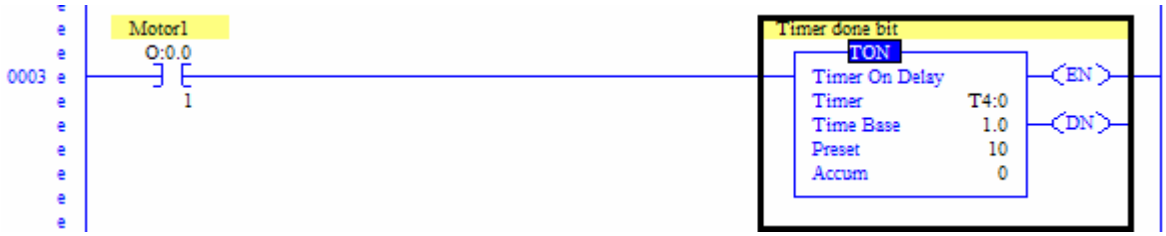
Vaš logički krug bi trebao izgledati ovako:



Sada ćemo dodati Timer naredbu.

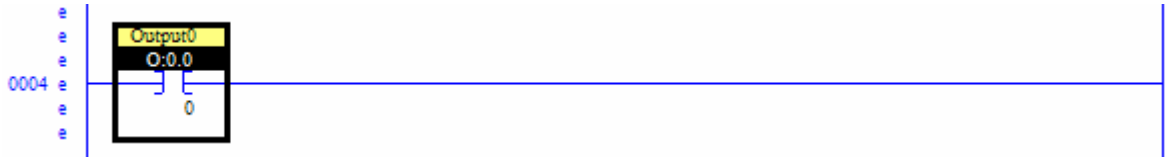
4. Kliknite na **Timer/Counter** tipku, koja se nalazi blizu vrha RSLogix 500 softvera. To mijenja tipke za naredbe u tipke za timere i brojače.
5. Kliknite, držite i vucite **TON** (Timer ON delay) tipku na desnu stranu logičkog kruga kojeg ste upravo projektirali.
6. Unesite sljedeće parametre timera, tako da dva puta kliknete na svaki parametar Timer naredbe.
   a. Timer: `'T4:0'` **Enter** (Ovo je adresa timera.)
   b. Time Base: `'1.0'` **Enter** (Timer broji jednom u sekundi.)
   c. Preset: `'10'` **Enter** (Vrijednost koju timer mora doseći da bi završio s radom.)
   d. Accum: `'0'` **Enter** (Trenutno stanje timera.)

Vaš bi logički krug sada trebao izgledati ovako:



6.3.5 Dodavanje brojača vašem programu ladder logike

1. Koristeći ono što ste naučili do sada, dodajte novi logički krug 4.
2. Zatim dodajte **XIC** tipku na lijevu stranu logičkog kruga kojeg ste upravo projektirali.
3. Da je naredba aktivna, upišite: `'O:0/0'`, **Enter.**
4. Vaš bi logički krug trebao izgledati ovako.



5. Kliknite na **Timer/Counter** tipku. To mijenja tipke za naredbe u tipke za timere i brojača.
6. Kliknite,držite i vucite **CTU** (CounT Up) tipku na desnu stranu logičkog kruga kojeg ste projektirali.
7. Unesite sljedeće parameter za brojač, tako da dva puta kliknete na svaki parametar naredbe brojača.

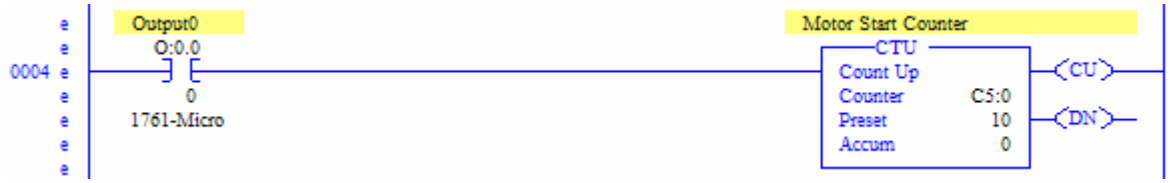
a. Brojač (counter): `'C5:0'` **Enter** (Ovo je adresa brojača.)

b. Ako vas se traži da unesete naziv, unesite `'Motor Start Counter'` i kliknite **OK**

c. Preset: `'10'` **Enter** (Vrijednost koju brojač mora doseći da završi s radom.)

d. Accum: `'0'` **Enter** (Ovaj broj pokazuje do kojeg je broja brojač trenutno nabrojao.)

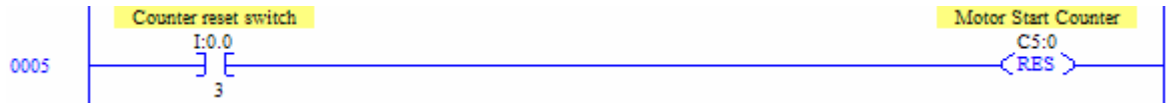
Vaš bi logički krug trebao izgledati ovako.



Sada trebamo dodati naredbu, kako bi mogli resetirati naš brojač. Kliknite na **User** tipku.

8. Dodajte novi logički krug 5.
9. Dodajte **XIC** tipku na lijevu stranu logičkog kruga kojeg ste upravo kreirali.
10. Upišite `'I:0/3'`, zatim **Enter**.
11. Ako vas se traži da unesete naziv, unesite `'Counter reset switch'` i kliknite **OK**.
12. Kliknite na **Timer/Counter** tipku. To mijenja tipke za naredbe u tipke za timere i brojače.
13. Kliknite i vucite **RES** tipku na desnu stranu logičkog kruga koje ste upravo kreirali. Ovo je RESet naredba koja nam omogućava da resetiramo naš brojač.
14. Upišite `'C5:0'` i **Enter**.

Vaš bi logički krug trebao izgledati ovako.



Funkcija RES ili reset naredba u našem programu je dovela 0 u akumulator našeg C5:0 brojača (engl. counter).

6.3.6 Dodavanje Limit instrukcije vašem programu ladder logike

1. Kliknite na **User** tipku.
2. Dodajte novi logički krug 6.
3. Kliknite na **Compare** tipku.

4. Kliknite i vucite **LIM** tipku na lijevu stranu logičkog kruga kojeg ste upravo projektirali. Ovo je LIMit test naredba. Omogučit će nam da usporedimo vrijednost našeg timera, sa unaprijed određenim granicama.
5. Unesite sljedeće parameter za LIM, tako da dva puta kliknete na svaki parametar LIM naredbe.

   a. Low Lim: `'3'` **ENTER** (Ovo je donja granica koju ćemo koristiti za našu usporedbu.)

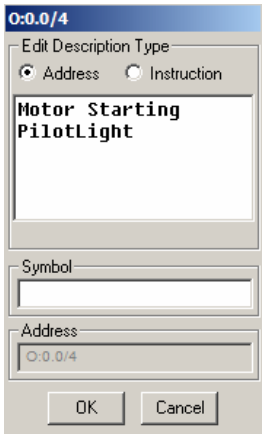
   b. Test: `'T4:0.ACC'` **ENTER** (Koristiti ćemo akumulator našeg timera kao vrijednost koju treba procijeniti.)

   c. High Lim: `'7'` **ENTER** (Ovo je gornja granica koju ćemo koristiti za našu usporedbu.)

> Opaska: LIMit naredba je istinita kada je timer između 3 i 7 sekunde. Znak pitanja će nestati kada pohranite program.

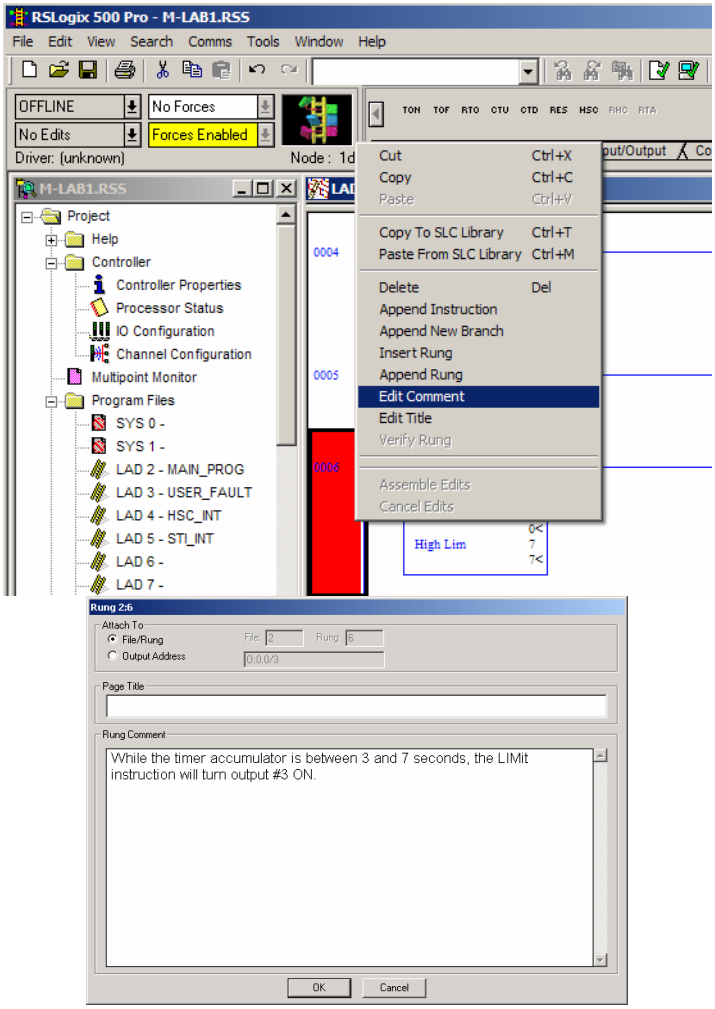
Dodajmo sada izlaznu naredbu našem LIM logičkom krugu.

6. Kliknite na **Bit** tipku.
7. Kliknite i vucite **OTE** tipku na desnu stranu LIM logičkog kruga kojeg ste upravo projektirali.
8. Upišite `'O:0/4',` zatim **Enter**.
9. Dajmo nazive našim novim izlaznim adresama.



6.3.7 Dodavanje komentara logičkom krugu

1. Kliknite na logički krug **6** (rung **0006**), kako biste istaknuli logički krug.
2. Kliknite desnu tipku miša na logičom krugu **0006**, i odaberite **Edit Comment.**



3. Odaberite **File/Rung.**
4. Upišite: `'While the timer accumulator is between 3 and 7 seconds, the LIMit instruction will turn output #3 ON'.`

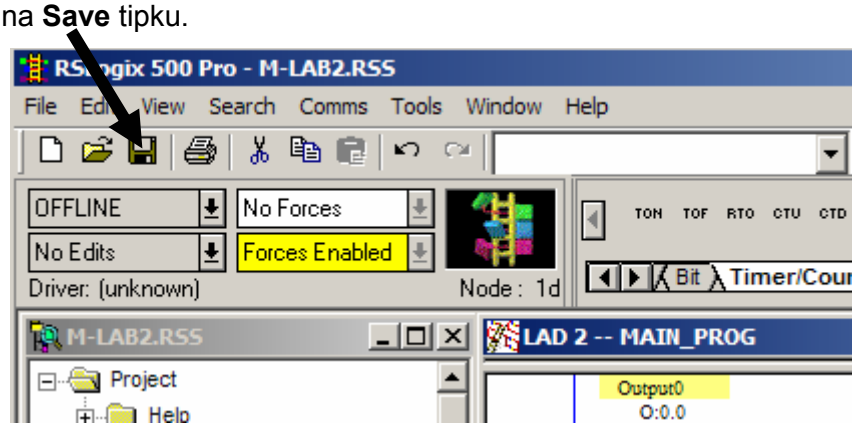
(“Dok je timer akumulator između 3 i 7 sekunde, LIMit naredba će uključiti izlaz #3.”)

5. Kliknite **OK.**

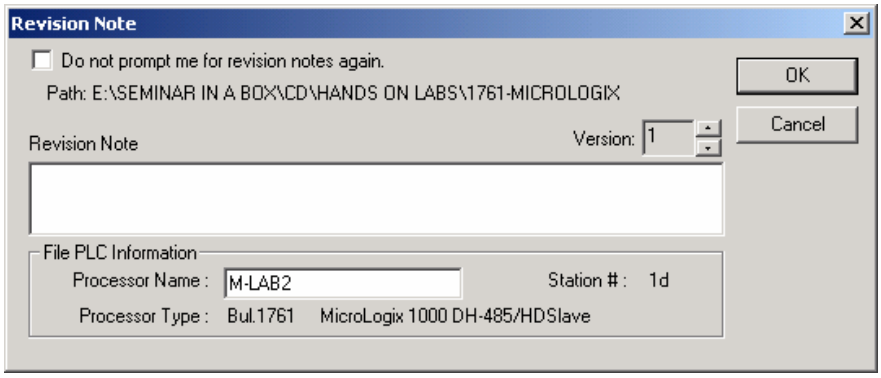
Komentari logičkog kruga se mogu koristiti kako bi se u detalje opisale funkcije logičkog kruga ladder logike. To je dobra karakteristika RSLogix 500 programskog softvera.

6.3.8 Pohrana vašeg rada

1. Kliknite na **Save** tipku.

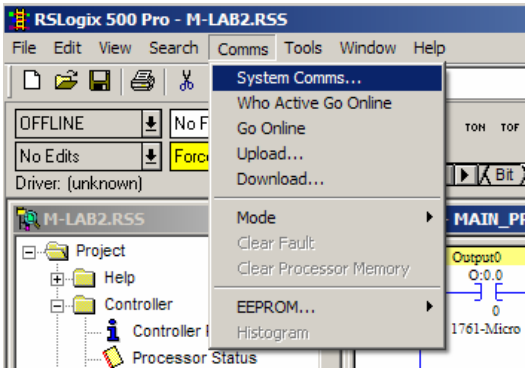


2. Kliknite **OK** kako bi otvorili Revision note prozor:



6.3.9 Download programa ladder logike u PLC-u

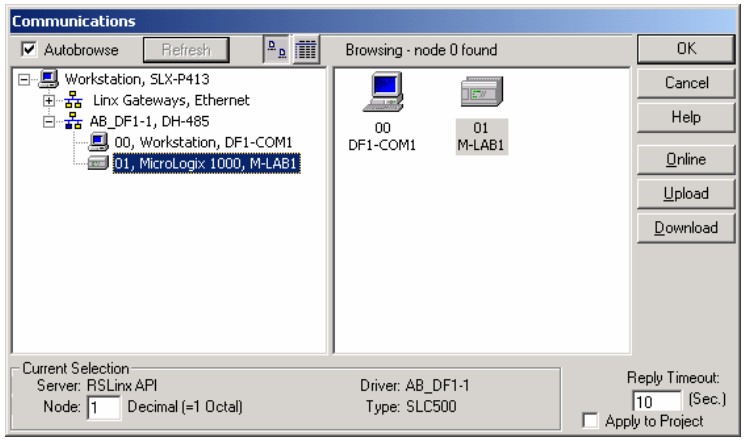
1. Iz menija **Comms** → odaberite **System Comms.**



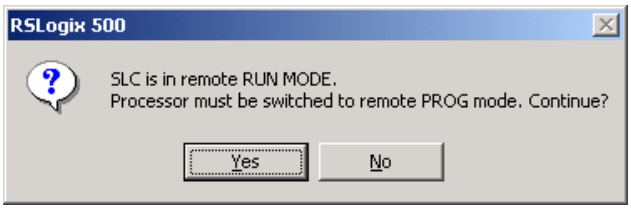
Primjetiti ćete da se ondje nalaze tri primarna odabira:

a. “Online”utvrđuje“put”.
b. “Upload” prima od kontrolera.
c. “Download” šalje kontroleru.

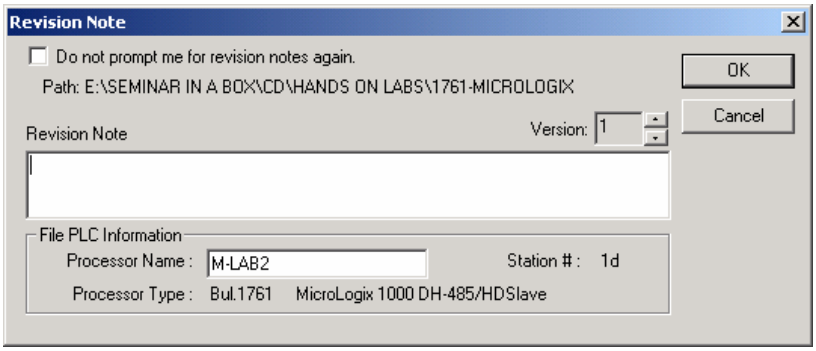
2. Istaknite uređaj kod Node **01**.
3. Kliknite **Download.**



4. Ako vidite prozor koji izgleda slično onome prikazanom dolje, kliknite: **Yes.**

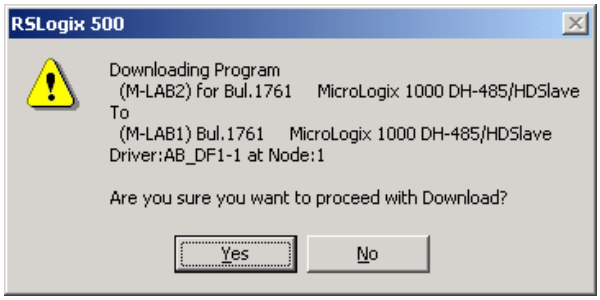


5. Odaberite **OK** u Revision note prozoru.



Opaska: Revision note karakteristika/značajka se može po volji isključiti, tako da se klikne na X u gornjem desnom kutu.

6. Odaberite **Yes** kako bi snimili preko postojećeg programa koji se nalazi u procesoru.



7. Ako se od vas traži da se vratite natrag u run mod, kliknite **NO**.

Gornji prozor će se pojaviti, svaki puta kada se program snima (download) u processor, kako bi se potvrdile vaše namjere da želite učiniti upravo to.

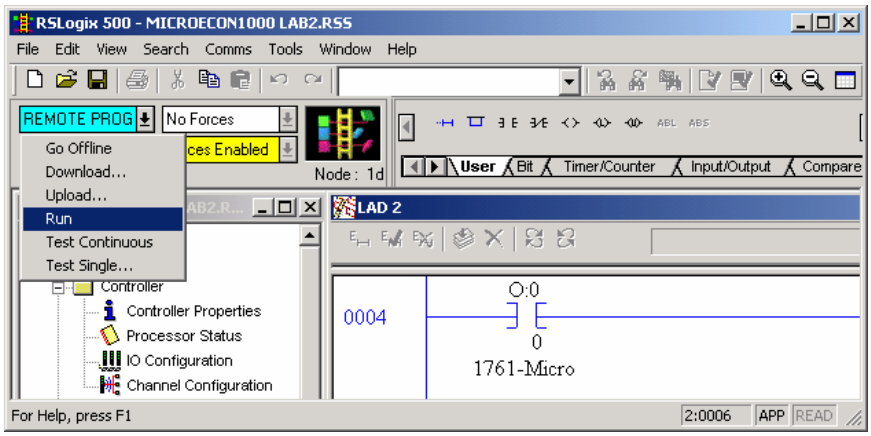
8. Odaberite **Yes** kako bi bili: online. To će vam omogućiti da monitor program kojeg ste upravo snimili u vaš MicroLogix kontroler.
9. Kliknite na strelicu koja pokazuje prema dolje, a nalazi se pokraj prozora sa riječi: **REMOTE PROG.**

Primjetite da ima tri (3) Run odabira :

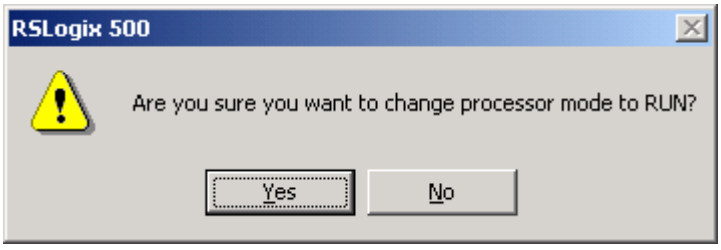
"Run" PLC 'scans' i izlazi u uključeni .

"Test Continuous" PLC 'scans' program i izlazi su isključeni.

"Test Single Scan" PLC izršava skeniranje jednog ciklusa, sa isključenim izlazima



10. Odaberite **Run.**
11. Nakon što se otvorio prozor "Are you sure?" ("Jeste li sigurni?"), kliknite **Yes.**



6.3.10 Nadziranje i testiranje vašeg programa ladder logike

1. Koristeći sklopke (engl. switch-eve) i promatrajući svjetla (engl. lights), da li vaš program radi onako kako ste očekivali?

Ispitajmo kako se ponaša vaš logički krug!

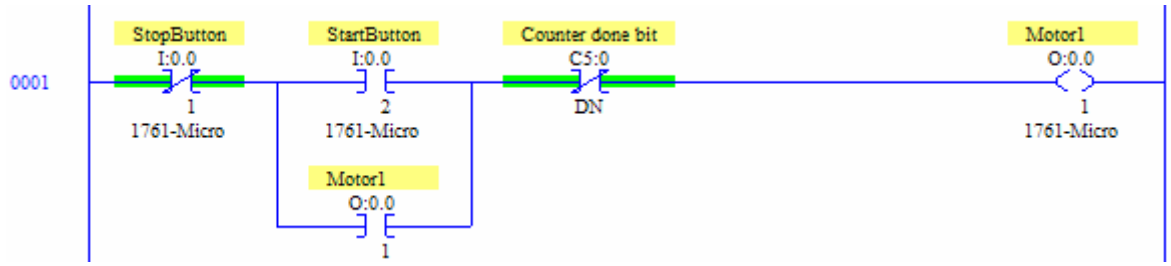
2. **Logički krug 0 (RUNG 0)**

Uključuje input switch #0 na input simulatoru i istražuje prednju stranu MicroLogix kontrolera. Output #0 će se uključiti, i u vašem logičkom krugu ćete vidjeti 'InputSwitch0' i 'Output0' aktivne, što znači da su input switch i output bili uključeni. Ako isključite 'Inputswitch0', tada će se 'Output0' isključiti.

3. **Logički krug 1 (RUNG 1)**

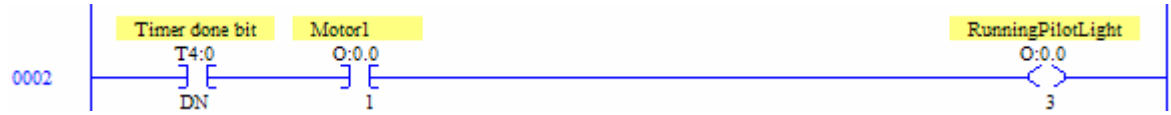
Pažljivo promatrajući vaš logički krug, uključite, zatim isključite 'StartButton' ili treći prekidač input simulatora. Uočite da su se 'Motor1' i 'Output1' uključili. Također uočite da je na trećem logičkom krugu brojač počeo brojati, zato što je njegov ulazni uvjet 'Output1' bio uključen.

Čak i nakon što isključite 'StartButton' motor ostaje uključen, i timer nastavlja s brojanjem. Ovo je odličan primjer 'zaključanog' strujnog kruga (engl. 'latching circuit'), također nazvanog kontrolni krug sa tri žice (engl. 3-wire control circuit). Uključite, zatim isključite 'StopButton' (drugi prekidač-switch) na input simulatoru. Sada vidite da se 'Motor1' isključio i da je timer prestao s brojanjem.



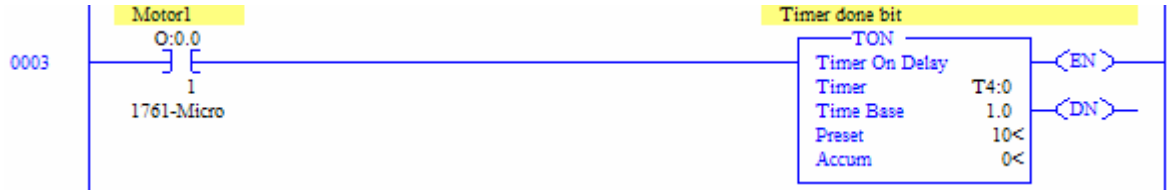
4. **Logički krug 2 (RUNG 2)**

Dva ulaza logičkog kruga 2 su izlazni uvjeti logičkog kruga 1 i 4. Izlaz 'Runningpilotlight' je uvjet za ova dva ulaza. Pa svaki od ovih izaza mora biti istinit ,kako bi uključio naš 'Runningpilotlight'. Da bi se to dogodilo, timer treba brojati 10 sekundi, tada će se output #2 uključiti. Jednom kada je motor uključen najmanje 10 sekundi, pojavit će se 'trčeće' svjetlo (engl. Running Pilot Light).



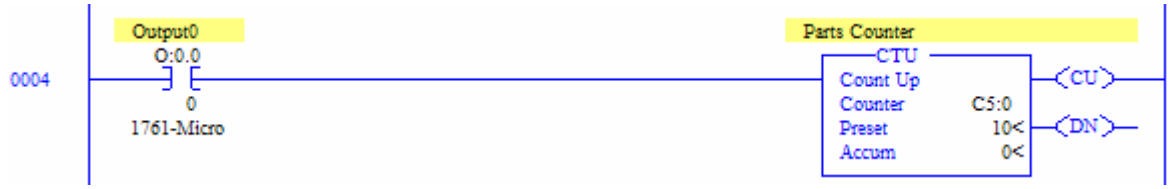
5. **Logički krug 3 (RUNG 3)**

Da bi TON (Timer On) počeo mjeriti sekunde, 'Motor1' mora biti uključen. Timer počinje s radom u isto vrijeme kada motor počne raditi.



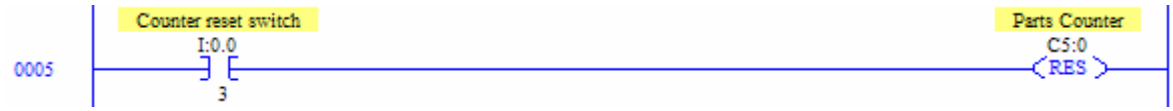
6. **Logički krug 4 (RUNG 4)**

Svaki put kada se 'Output0' promijenilo stanje iz isključen u uključen, naš brojač C5:0. će se uvećati za 1. To će nam omogućiti da pratimo koliko puta je motor bio uključen, i adekvatno tome kada se treba obaviti kontrolni pregled.



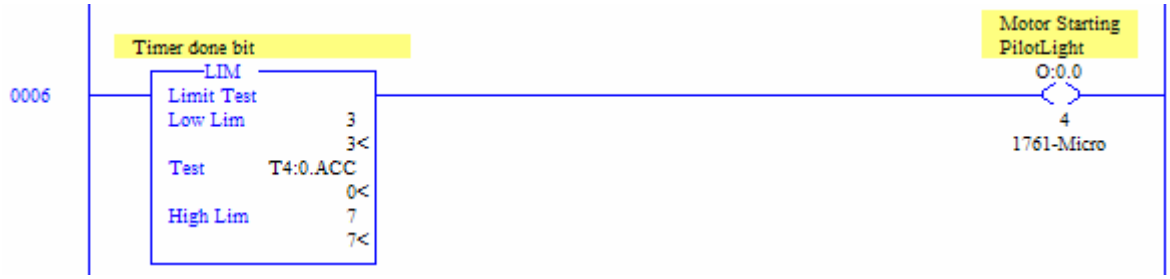
7. **Logički krug 5 (RUNG 5)**

Kada se naš brojač resetira, sklopka će se uključiti, tj. brojač C5:0 će resetirati natrag na 0. To će također ponovno uključiti output #1 (izlaz #1) i timer. Kada se radi održavanje motora, možemo resetirati brojač kako bi mogli pratiti kada je drugi kontrolni pregled.



8. **Logički krug 6(RUNG 6)**

Kada je akumulator timera između 3 i 7 sekunde, naredba LIMit će uključiti output #4 (izlaz #4). To nam ukazuje da je motor započeo s radom, i tada se isključiti kako bi sačuvao vrijeme života našeg vođenog svjetla (engl. pilot light).



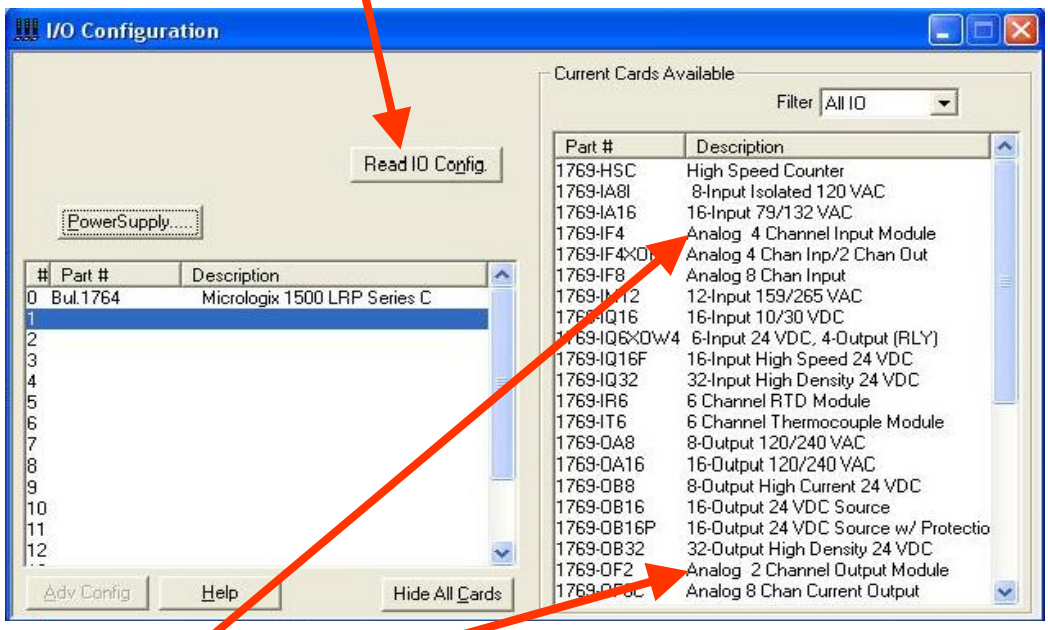
6.4 Vježba 4

6.4.1 Konfiguracija analogih ulaza i izlaza

> Opaska: Da bi analogni ulazi i izlazi PLC-a ispravno funkcinirali potrebno je obaviti konfiguraciju istih

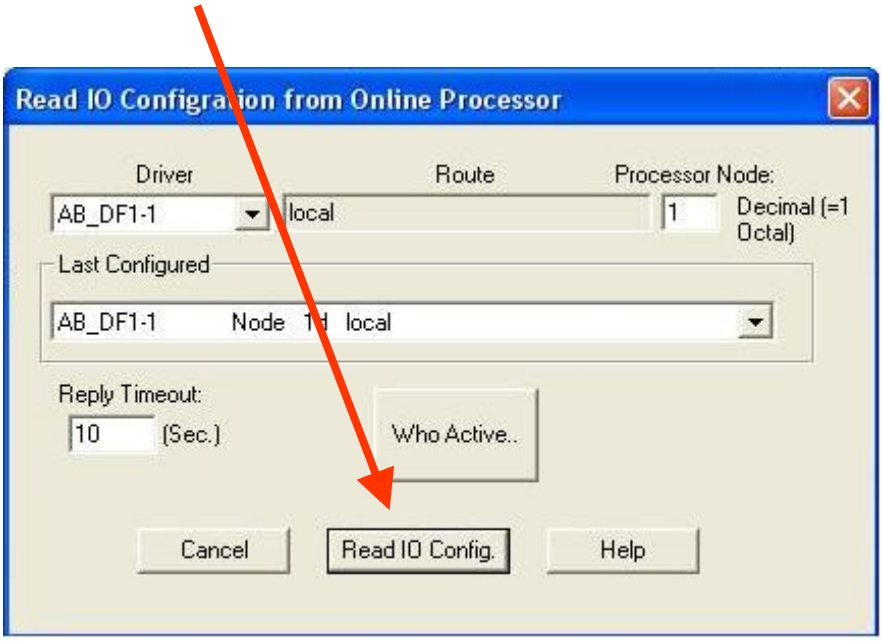
Funkcije za odabir dodatnih modula s analognim ulazima i izlazima te njihovu konfiguraciju nalaze se na lijevoj strani prozora RSLogix-a.
Da bi odredili module priključene na PLC i konfigurirali AI (analog input) i AU (analog output) na njima potrebno je:

1. Odabrati **IO Configuration** (na lijevoj strani prozora).
2. Nakon odabira, otvorit će se prozor **IO Configuration** u kojemu možemo odredili module priključene na PLC i obaviti konfiguraciju naših AI i AU.
3. Kliknemo na **Read IO Configuration,** tako da dobijemo novi prozor u kojemu odredimo sa kojeg uređaja (PLC-a) čitamo AI i AU

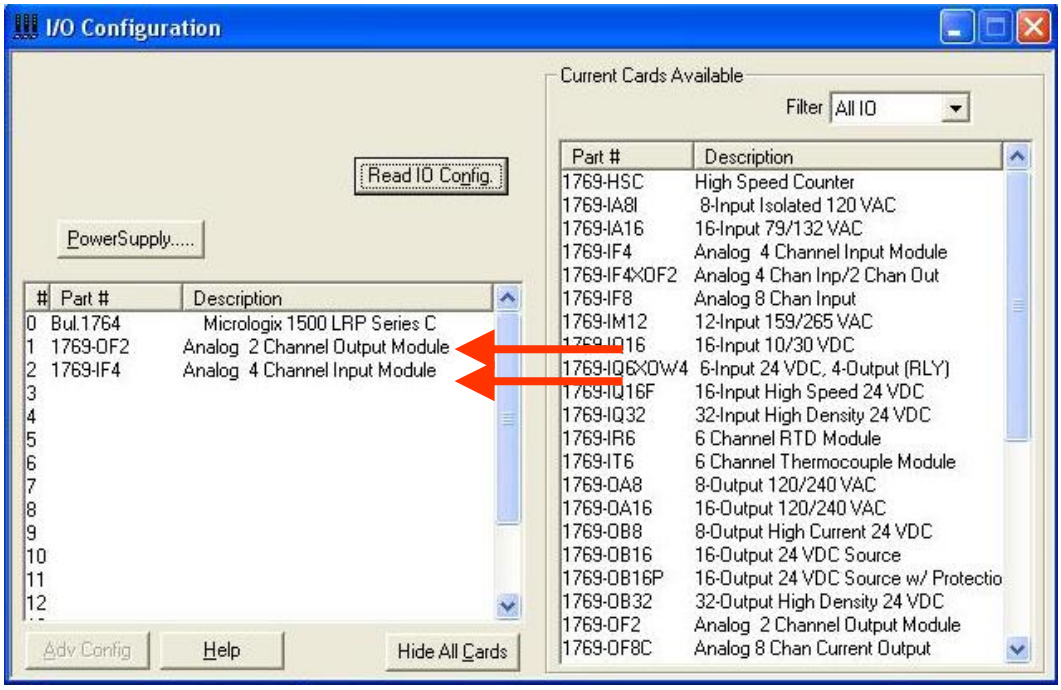


> Opaska: Module priključene na PLC možemo i ručno odabrati, tako da ih po nazivima nađemo na desnoj strani **IO Config**, kliknemo lijevom tipkom miša i držimo ju pritisnutu, te povučemo na lijevu stranu u prazan stupac. Moduli moraju biti pod brojem po redosljedu kojim su spojeni na PLC.

4. U novom prozoru potrebno je odabrati vezu po kojoj pristupamo našem PLC-u i kliknuti na tipku **Read IO Config**.



5. Vraćamo se u prozor **IO Configuration**, te nam se automatski iščitaju postojeći moduli za proširenje trenutno spojeni na PLC i ispisani po redosljedu. Module odnosno njihove AI i AO moramo podesiti prema zahtjevima sustava (u protivnom neće raditi kako treba).



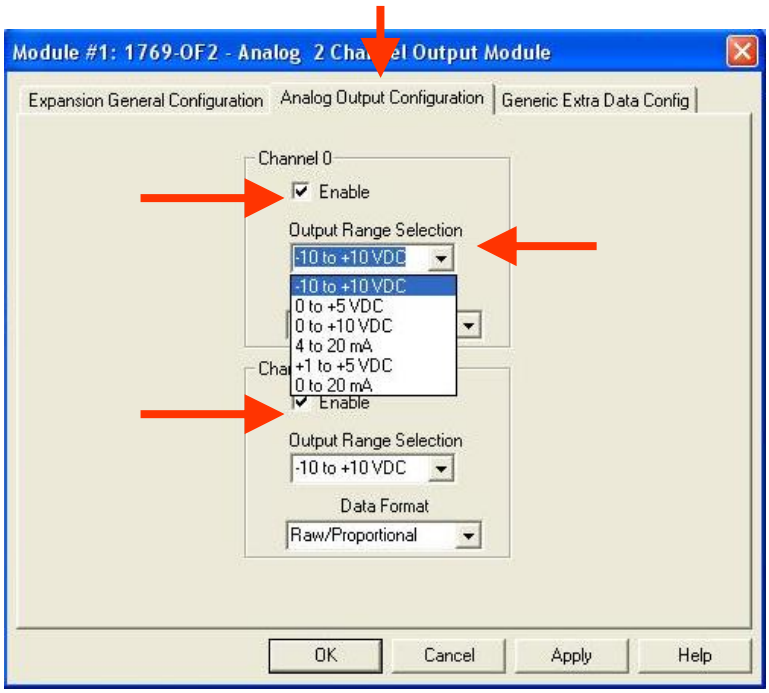
6.4.1.1 Pojedinačno podešavanje analognih ulaza i izlaza

Podešavanje svakog pojedinog analognog ulaza i izlaza vršimo u prozoru **IO Configuration**, na slijedeći način:

1. Dva puta kliknemo na tekst imena modula za proširenje u lijevoj strani prozora, tako da dobijemo novi prozor (u našem smo slučaju odabrali analogni izlaz **Analog 2 Channel Output Module**).
2. Odabiremo karticu sa nazivom **Analog Output Configuration** i omogućujemo rad AO tako da stavljamo kvačicu u kvadratić gdje piše **Enable** pod **Channel 0** (O:1.0) i **Channel 1**
(O:1.1).
3. U ovom prozoru također je potrebno odrediti **Range** (izlazni opseg)**,** tako da iz padajućeg izbornika odaberemo vrijednost za opseg sa kojim mi želimo raditi (u ovoj vježbi odabiremo -10 to +10 VDC).

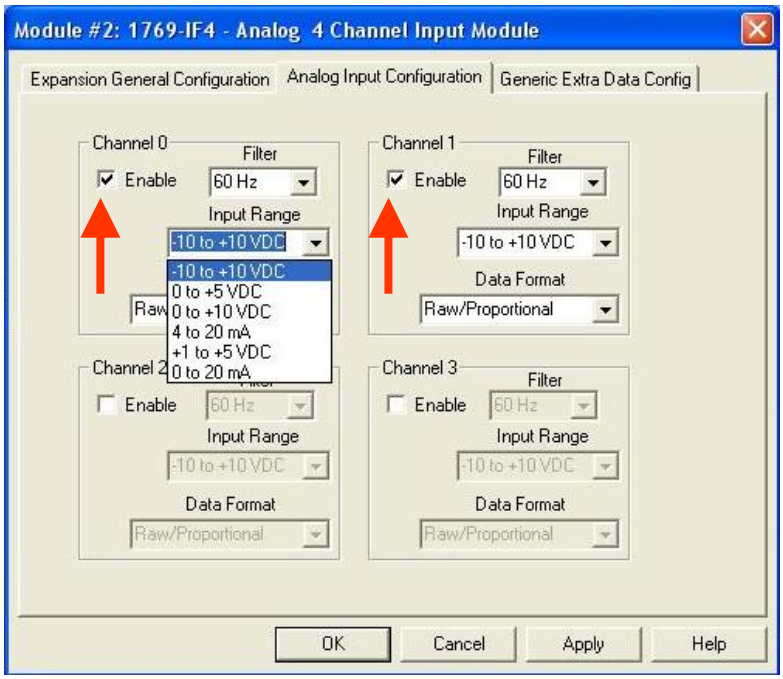
Opaska: U slučaju AO to je izlazna strujna ili naponska veličina koju šaljemo u sustav i njome upravljamo izvršnim uređajima (položaj ventila, broj okretaja motora, itd). U slučaju AI to je strujna ili naponska veličina koju dobivamo s senzora iz proizvodnog procesa.

4. Pošto smo podesili parametre AO, kliknemo tipku **Apply**, pa potom tipku **Ok**



Opaska: Važno je uvijek pravilno podesiti **Range** jer u protivnom može doći do oštećenja uređaja

5. Isti proces ponavljamo za analogne ulaze (**Analog 4 Channel Input Module**)



Opaska: Od četiri kanala koja se pojavljuju na izbor, mi odabiremo samo Channel 0 (I:2.0) i Channel1 (I:2.1), a ostala dva ne aktiviramo zbog toga što na ovoj vježbi koristimo samo prva dva (ovdje također potrebno odrediti **Range** i to -10 to +10 VDC)

6. Ako smo sve dobro odradili iz prijašnjih točaka, imamo podešene naše AI i AO, pa možemo izaći iz prozora IO Configuration.

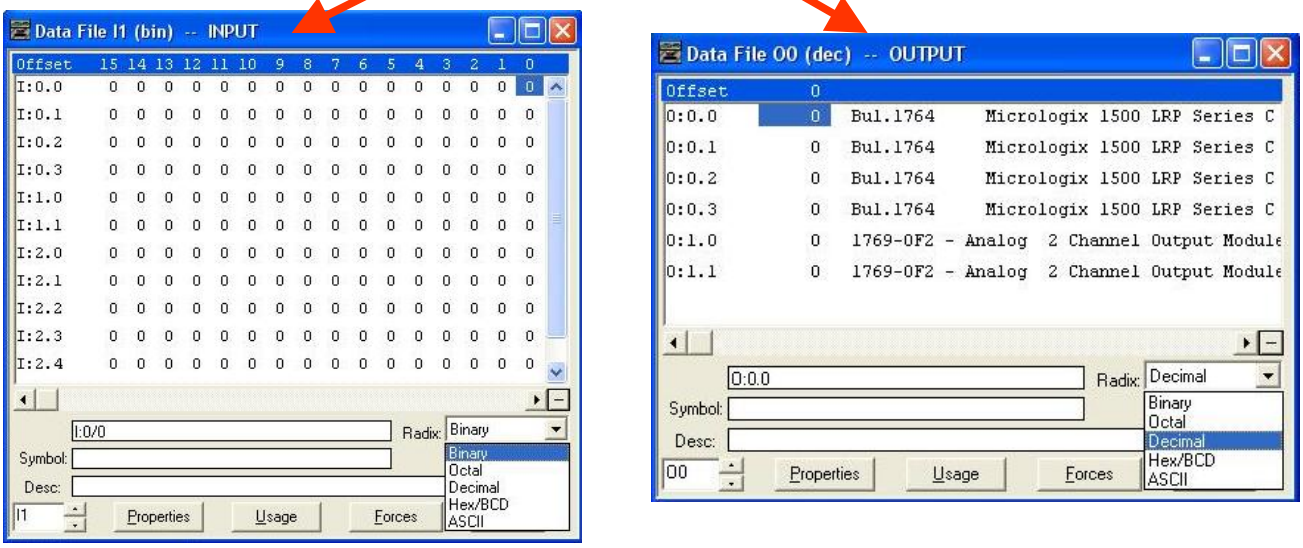
Opaska: Pojedine tipove PLC-a nije potrebno dodatno podešavati iz razloga što su fiksne konfiguracije AI i AO, ali je svakako potrebno odrediti koji su moduli za proširenje spojeni na uređaj.

6.4.2 Kontrola i pregledavanje trenutnog stanja AO i AI

Kontrolu i pregledavanje trenutnog stanja AO i AI potrebno je izvršiti da bi mogli provjeriti kako i što naš PLC radi, (da li radi ono što smo mi očekivali od njega, što ćemo dobiti na izlazu, da li smo dobro podesili ulaze, itd).

Kontrolu AI i AO možemo obavljati tako da:

1. U prozoru sa lijeve strane nađemo izbornik **Data file** (isti izbornik gdje smo našli **IO Config**.). Iz njega izaberemo **I1 Input** ili **O0 Output**, ovisno o tome što želimo, te dva puta kliknemo na odabrani.



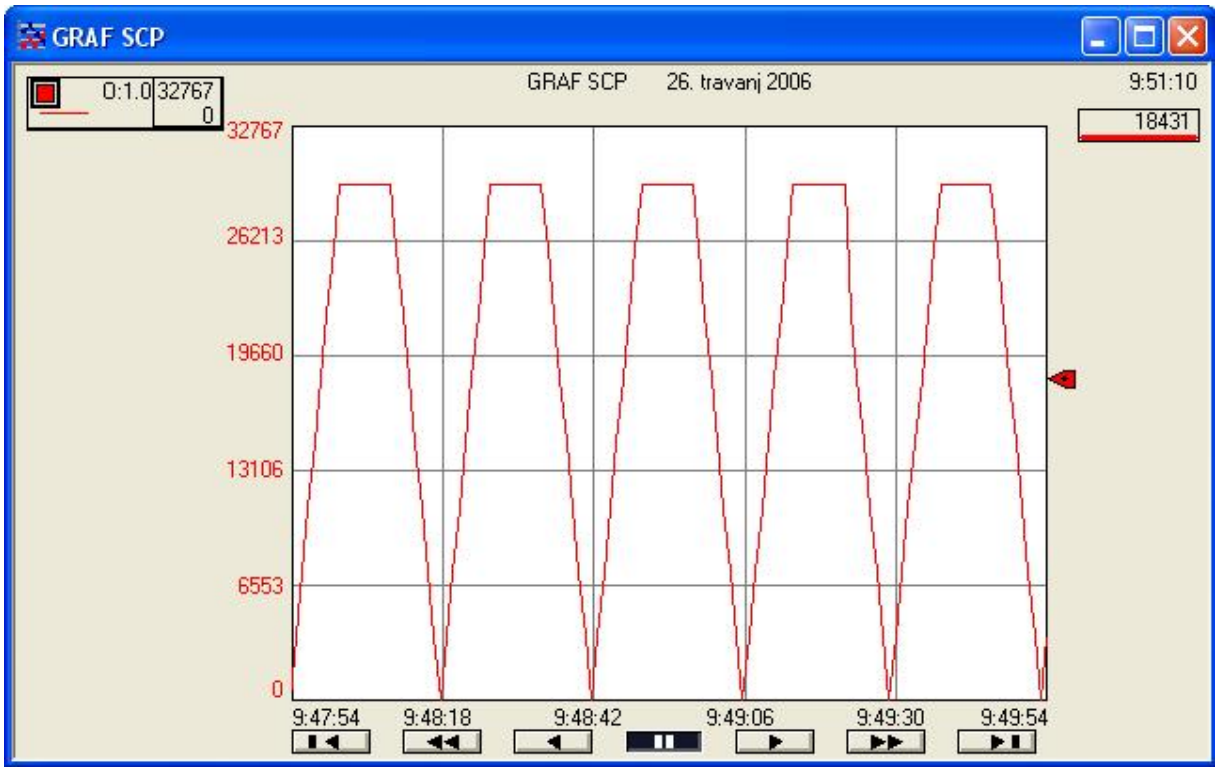
2. Pošto smo otvorili prozor **Data File** možemo kontrolirati naše **AO ili AI** , na način da očitavamo vrijednosti koje dobijemo na **O/I** i usporedimo ih sa našim željenim (očekivanim) vrijednostima, npr. pratimo I:0.0 i O:0.1 ili I:1.0 i O:0.0, itd.
3. Vrijednosti koje promatramo možemo dobiti u obliku koji nam je lakše promatrati. Prema početnim postavkama vrijednosti su uvjek prikazane u binarnom stanju (**Binary**), ali ako kliknemo na opciju **Radix,** otvorimo padajući niz i izaberimo jednu od opcija - **Octal, Decimal**, **Hex/BCD** ili **ASCII**.

Naše vrijednosti AI i AO možemo promatrati u grafičkim prikazu, gdje su vrijednosti odabranih O/I prikazani u ovisnosti o realnom vremenu, na slijedeći način:

1. Graf otvaramo tako da u našem izbornik na lijevoj strani pronađemo opciju **Trend**, te na nju kliknemo desnom tipkom miša
2. Otvara nam se opcija **New** na koju kliknemo lijevom tipkom miša, tako da dobijemo prozor **Add Trend** gdje upisujemo ime garfa po želji i kliknemo na **Ok** ili pritišćemo **Enter** tipku na tastaturi.
3. Ispod opcije **Trend**, naći ćemo naziv tog grafičkog prikaza kojeg smo u prethodnoj točci nazvali po želji, a njega otvaramo tako da dvaput kliknemo lijevom tipkom miša na ime grafa.

Opaska: Graf koji smo dobili prikazuje nam kako se mijenjaju vrijednosti AI i AO u odnosu na realno vrijeme. Vrijeme nam je ispisano na osi X, a na osi Y se ispisuje vrijednost analognog O/I kojega želimo promatrati u ispisu na grafu

Na grafu prikazanom na sljedećoj slici pokazan je primjer rada motora (broj okretaja motora prilikom faze pokretanja, stalnog rada i zaustavljanja) i to sve u odnosu na realno vrijeme u kojem se taj proces odvijao:



Napomena:

Prije samog pokretanja PLC-a u **RUN mode** potrebno je navesti izlaz koji želimo promatrati na grafu, a to ćemo izvršiti na slijedeći način:

1. U prozoru sa lijeve strane nađemo izbornik **Data file** (isti izbornik gdje smo našli **IO Config**.), iz njega izaberemo **I1 Input** ili **O0 Output**, ovisno što želimo, te dva puta kliknemo na odabrani
2. Kliknemo na opciju **Radix,** otvorimo padajući niz i izaberimo **Decimal**.
3. Kliknemo i držimo lijevu tipku miša na Output O:1.0 - koji smo naveli u programu (u funkciji koju koristimo, npr. SCP) i povučemo u naš graf.
4. Kada dođemo do našega prozora u kojem se nalazi graf pustimo tipku.
5. Ako smo dobro ovo napravili pojaviti će nam se naš Output na lijevoj gornjoj stani grafa, pa možemo krenuti sa pokretanjem **Run moda** i očitavanja vrijednosti sa grafa

Opaska: Mjerne skale koje nam se nalaze na osima možemo sami mijenjati po želji, nije obavezno koristiti vrijednosti koje nam se automatski ispisuju na osima grafa

Mjerne skale mjenjamo na slijedeći način :

1. Pomoćni izbornik otvaramo klikom desne tipke miša na samu sliku našeg grafa
2. Odabiremo **Chart Properties,** na koji ćemo jedanput kliknuti lijevom tipkom miša, da nam se otvori novi prozor **RSTrendX Properties**
3. U **RStrendX Properties** za promjenu mjerne skale **Y-osi**, odabrat ćemo karticu s nazivom **Y-Axis**
4. Potražiti opciju **Custom** koju otvaramo, tako da nam se linije za min. i max veličine zabijele i u njih upisujemo vrijednosti koje želimo da nam se prikažu na Y osi našeg grafa (npr. za AI ili AO koji imaju opseg -10 do +10 VDC treba upisati min -32767, a max 32767)
5. Kliknemo na **Ok** tako da potvrdimo promjenu i da se vratimo na graf
6. Isto tako vrijednosti mjerne skale možemo mjenjati na još jeden način, ako kliknemo lijevom tipko miša na samu Y-os (držimo je pritisnutu) i vućemo je u smijeru kojem želimo (prema gore ili prema dolje), mijenja nam se vrijednost skale

Opaska: Precizniji način za mijenjanje mjerne skale svakoko je korištenje pomočnog izbornika i naredbe **Custum,** jer se upisuje točan broj maksimuma kojega zelimo na grafu

6.4.3 Rad s naredbama komparacije

6.4.3.1 Uvod

Da bi uspješno radili nad podacima dobivenim s analognih ulaza i izlaza, trebalo bi ponoviti postupak konfiguracije uređaja. U poglavlju 6.4.1. se opisuje podešavanje sklopovske konfiguracije uređaja.

> Opaska: PLC Micrologix 1500 LRP Series C ima 2 analogna ulaza koja se deklariraju kao I:2.0 i I:2.1, te 2 analogna izlaza koji se deklariraju kao O:1.0 i O:1.1. (I - Input (ulaz), O - Output (izlaz)).

Uređaj u radu s analognim ulazima ili izlazima može raditi prema našim potrebama sa standardnim naponskim ili strujnim veličinama prikazanim u tablici.

	RASPON RADA
ANALOGNI ULAZI (I:2.0 i I:2.1) i **ANALOGNI IZLAZI** (O:1.0 i O:1.1)	od -10 do +10 VDC
	od 0 do + 5 VDC
	od 0 do +10 VDC
	od 4 do 20 mA
	od + 1 do + 5 VDC
	od 0 do 20 mA

Prilikom programiranja, korisnik ne radi direktno s naponskim (V) i strujnim veličinama (mA), već se podaci zbog A/D i D/A pretvorbe skaliraju prema tablici.

	SKALIRANJE	MOGUĆE LJESTVICE
ANALOGNI ULAZI (I:2.0 i I:2.1) i **ANALOGNI IZLAZI** (O:1.0 i O:1.1)	Raw/Proportional Data	od -32767 do +32767 (cijeli brojevi - ovisi o rezoluciji A/D pretvarača)
	EngineeringUnits	od -10500 do +10500
	Scaled for PID	od 0 do +16383 (cijeli brojevi) (može i van te skale, što ovisi o ulaznom kontroleru, za te vrijednosti treba obično pogledati u dobiveni priručnik)
	Percent Value[1]	pretvara veličinu u postotke

U radu s analognim U/I potrebno ih je prilagoditi potrebama, tj. uvjetima rada uređaja u pogonu. Prvo odabiremo naponski ili strujni opseg, a zatim i odgovarajuću skalu.

[1] pretvara sve u postotke npr. -0.5V to 10.5V = -5% to 105%; pojavljuje se samo ako nam raspon rada nije od –10 do +10 VDC

Primjer.

Podesimo analogni ulaz da radi u rasponu (**0 to +10 VDC**), i odaberemo ljestvicu cijelih brojeva od 0 do +32767 (**Raw/Proportional Data**). To znači da će on podjeliti raspon od 0 do +10V na 32767 djelova od kojih svaki odgovara točno određenom naponu unutar odabranog raspona (rezolucija je 10VDC/32767).

Opaska: Sve matematičke naredbe prilagođene su rasponu rada U/I od -10 do +10 VDC (**-10 to +10 VDC**) te ljestvici cijelih brojeva (**Raw/Proportional Data**) od -32767 do +32767. (Pogledati donju tablicu za primjer)

NAPON (V)	ODGOVARAJUĆI CIJELI BROJ
-10	-32767
0	0
5	16384
10	+32767

6.4.3.2 Naredbe komparacije

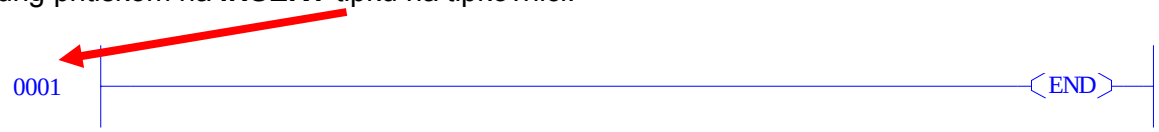
Naredbe komparacije su uvjetne naredbe u programiranju i obavljaju usporedbu nad vrijednostima određenih veličina u PLC-u (npr. uspredba AI (I:1.0) s konstantom upisanom u memorijski registar za cjelobrojne veličine (N7:1)). PLC Micrologix 1500 LRP Series C koristi 8 naredbi komparacije: granični uvjeti (**LIM**), maskirana jednakost (**MEQ**), jednakost **(EQU**), nejednakost (**NEQ**), manji (**LES**), veći (**GRT**), manji ili jednak (**LEQ**) i veći ili jednak (**GEQ**). Naredbe komparacije nisu ništa drugo nego ispitivanje točnosti logičkih tvrdnji (npr. da li je neka varijabla veća od neke druge i sl.)

Opaska: Naredba komparacije **ne smije** stajati sama u logističkom krugu.

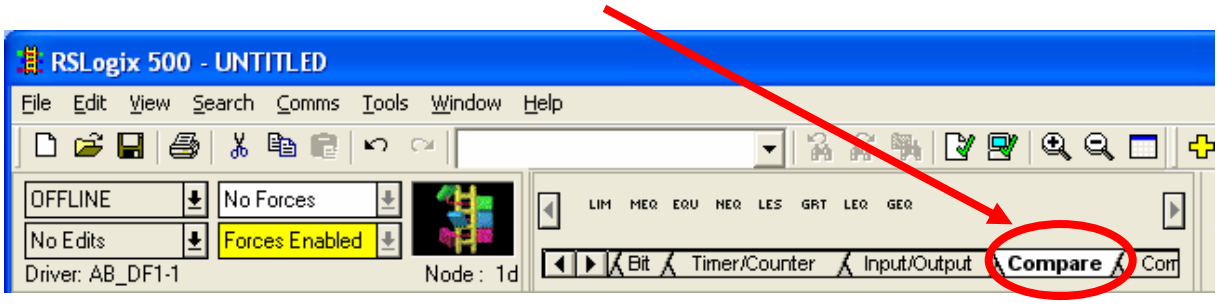
6.4.3.3 Postavljanje naredbe komparacije na rung

Naredbe komparacije se postavljaju na lijevu stranu rung-a i to na sljedeći način:

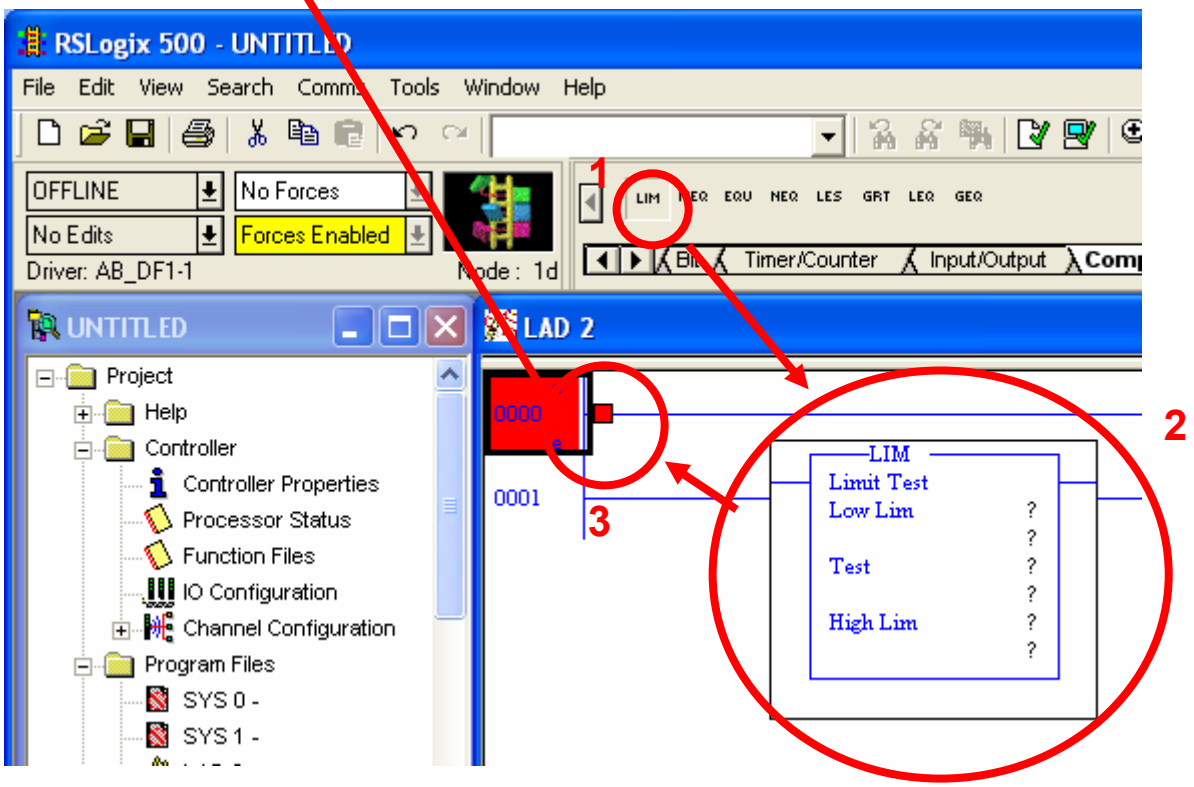
1. Pritiskom lijeve tipke miša odaberite početak END rung-a i dodajte jedan slobodan rung pritiskom na **INSERT** tipku na tipkovnici.



2. Odaberete karticu sa nazivom **Compare**.

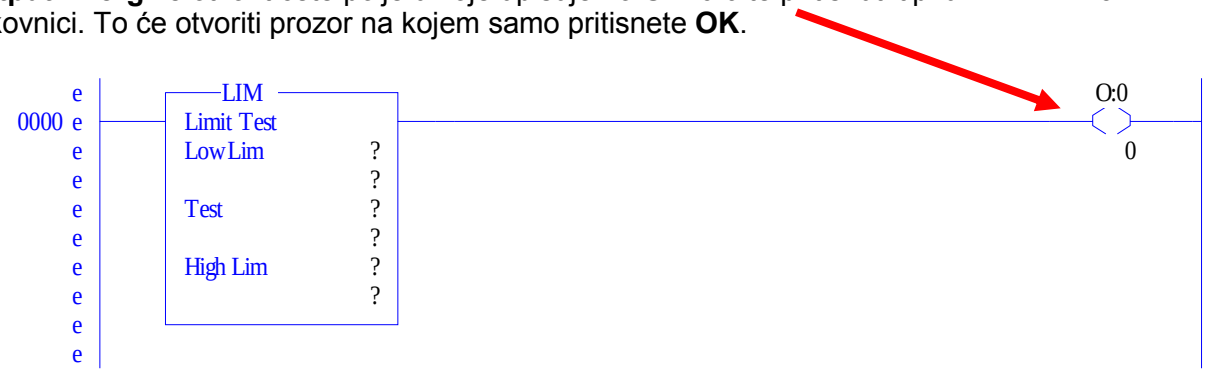


3. Zatim odaberete željenu naredbu (npr. **LIM**) pritiskom lijeve tipke miša na naredbu(**1**) i držite je pritisnutu dok prenosite kursor miša na početak runga (**3**) gdje je pustiti tek kad kvadratić pozeleni. (**LIM** prozor će se pojaviti tek kada kursor miša pređe na **LAD 2** prozor (**2**))



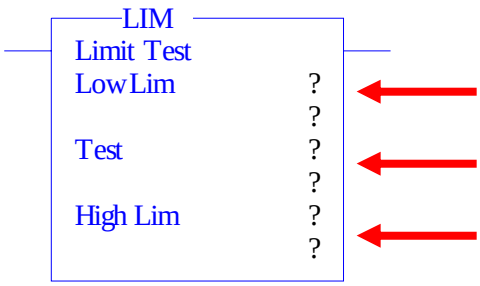
4. Na isti način dodati naredbu **Output Energize** koja se nalazi na kartici **User** na desnu stranu rung-a tako da sve izgleda kao na slici.

5. Dvostrukim pritiskom na lijevu tipku miša dok je kursor na upitniku naredbe **Output Energize** otvorit ćete polje u koje upisujemo **O:1.0/0** te pritisnuti tipku **ENTER** na tipkovnici. To će otvoriti prozor na kojem samo pritisnete **OK**.

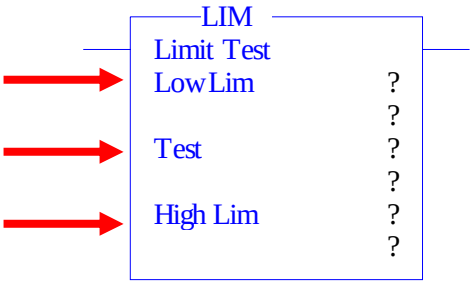


6. Na jednak način kako ste promjenili polje naredbe **Output Energize** mijenjaju se i polja u prozorima naredbi komparacije. Dovoljno je samo pritisnuti dvaput lijevu tipku miša iznad upitnika polja koje se želi izmjeniti.

7. Polja koja se mogu izmjeniti nalaze se pod upitnicima pokraj njihovih imena.

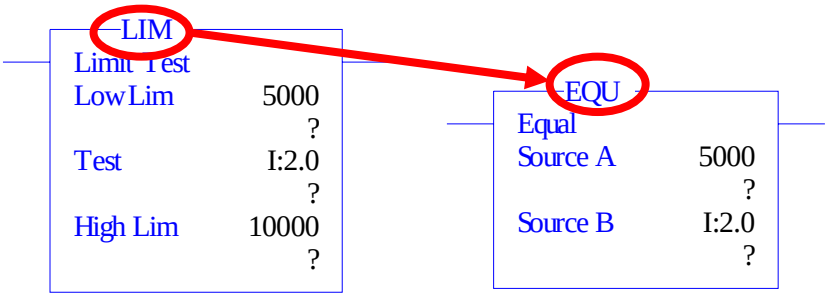


8. Upitnici koji se nalaze ispod izmjenjivih polja prikazuju trenutno stanje varijable (ili konstante) koja upisana u izmjenjivo polje iznad njih. (npr. ako u **Low Lim** polje upišemo **I:2.0**, tada će za vrijeme **REMOTE RUN** načina rada programa RSLogix 500 upitnik ispod prikazivati trenutno stanje analognog ulaza).



9. Još jedan način kako ″dodati“ prozor neke naredbe komparacije jest iskoristiti postojeći. Dvostrukim pritiskom na lijevu tipku miša na natpis naredbe komparacije (npr. **LIM**) otvorit će se polje u koje možete upisati ime neke druge naredbe

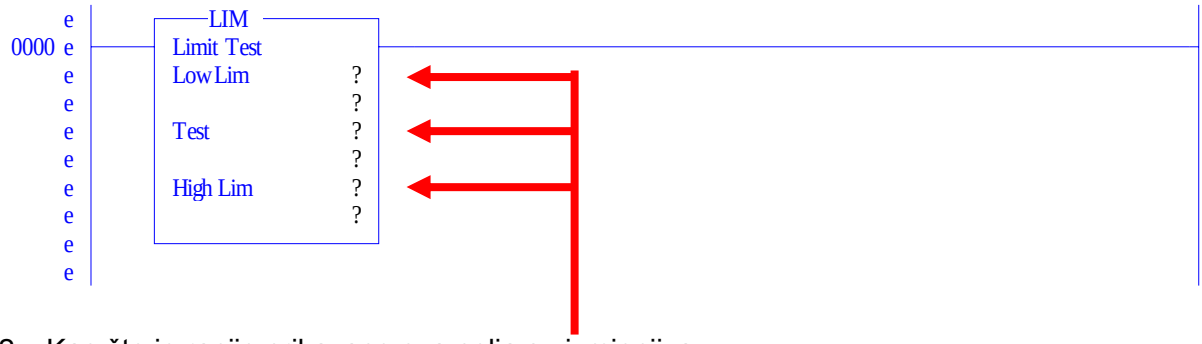
komparacije (npr. **EQU**) ili neke druge naredbe općenito i samo pritisnete tipku **ENTER** na tipkovnici nakon upisivanja.



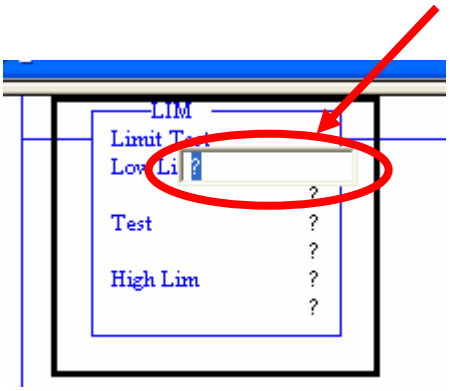
6.4.3.4 Polja za upisivanje unutar naredbenih prozora

Unutar svakog naredbenog prozora postoje neka polja koja je potrebno ispuniti kako bi PLC znao što mora računati, uspoređivati ili nešto treće. Polja se mogu ispunjavati različitim varijablama ili konstantama, ovisno o potrebama u procesu gdje se uređaj koristi. Za primjer ćemo uzeti komparacijsku naredbu **LIM**.

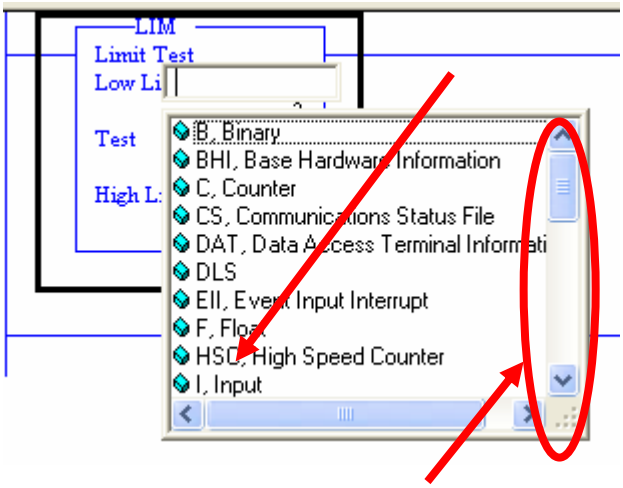
1. Postavite **LIM** naredbu na rung.



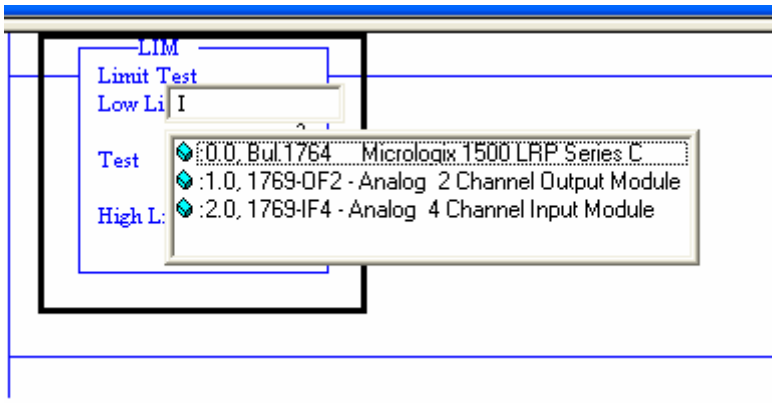
2. Kao što je ranije prikazano ova polja su izmjenjiva.
3. Dvostrukim pritiskom lijeve tipke miša na upitnik kod **Low Lim** otvorite polje za upisivanje.



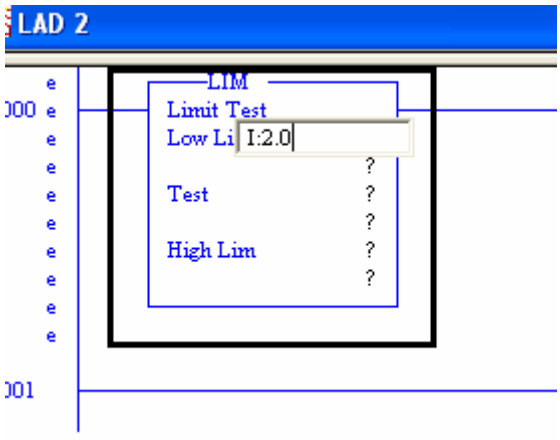
4. Dvostrukim pritiskom na tipku **Backspace** (ponegdje je označena dugom strelicom prema okrenuta lijevo) unutar novog otvorenog polja pojavit će se padajući izbornik.



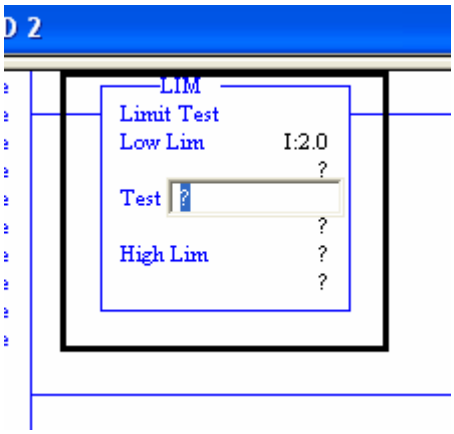
5. Po izborniku se možemo kretati uporabom kursora miša i slider-a ili strelicama na tipkovnici. Izbor je vaš. Odaberite **I, Input** i pritisnite **ENTER** tipku na tipkovnici. Pojavit će se sljedeći izbornik.



6. Zatim odaberite **:2.0, 1769-IF4 ...**.



7. Ako ste pritisnuli tipku **ENTER** nakon odabira, program vas direktno prebacuje u sljedeće polje. Dalje iskušavajte sami.

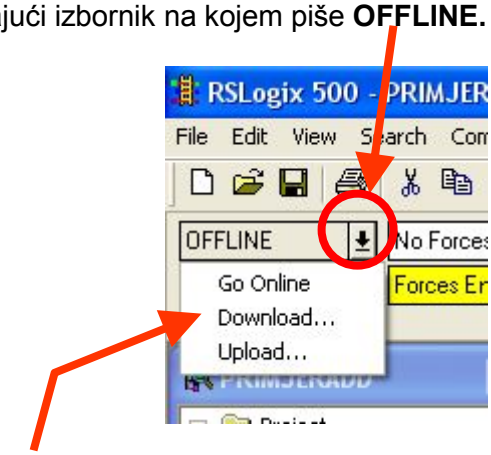


Opaska: Postoji mnogo različitih načina i kombinacija za upisivanje. Prikazan je najduži način, a najkraći način bi bio tako da za dani primjer nakon 3. koraka jednostavno upišete **I:2.0** i pritisnete tipku **ENTER** nakon unosa. Način unosa varijabli ili konstanti je vaš izbor, a na isti način se ispunjuju polja kod svih naredbi, bile one komparacijske, matematičke ili neke druge.

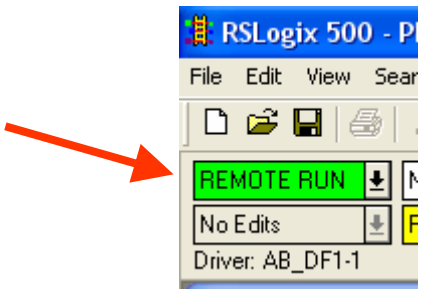
6.4.3.5 Prebacivanje programa na PLC

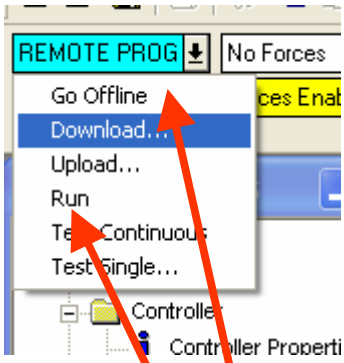
Kad ste napravili program koji želite da uređaj izvršava, napravite slijedeći postupak.

1. Kliknite na padajući izbornik na kojem piše **OFFLINE.**

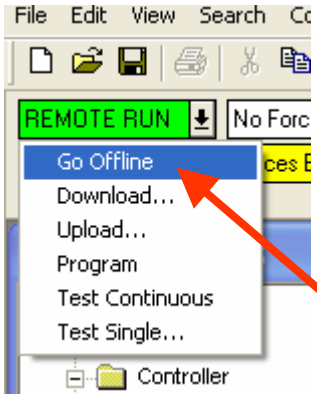


2. Odaberite **Download** i nakon toga na sve male prozore koji se pojave kliknite **Yes** sve dok se ne pojavi **REMOTE RUN** mod.



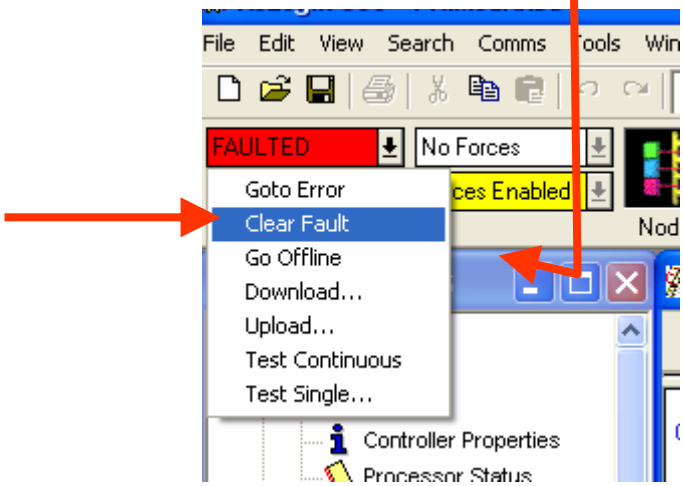


Opaska: U slučaju da se pojavi **REMOTE PROG**, sve što morate učiniti jest kliknuti lijevom tipkom miša na padajući izbornik u kojem se **REMOTE PROG** nalazi i odabrati **RUN**. **Samo** u **REMOTE RUN** načinu rada programa možemo promatrati stanje uređaja unutar programa RS Logix 500. MIJENJANJE POLJA TOKOM IZVOĐENJA PROGRAMA U **REMOTE RUN** NAČINU RADA JE **ONEMOGUĆENO** !!!!



Da bi ste mogli nastaviti programirati morate se prebaciti natrag na **OFFLINE** način rada.

Opaska: U slučaju da se kod izvodenja programa pojavi greška pojavit će se natpis **FAULTED** u prozorčiću u kojem je pisalo **REMOTE RUN** i **ne može nastaviti sa radom** sve dok se ta greška ne izbriše (**CLEAR FAULT**). U tom slučaju najbolje je prebaciti se na **OFFLINE** način rada i ispraviti grešku u samome programu.



6.4.3.6 LIM naredba (granični uvjeti)

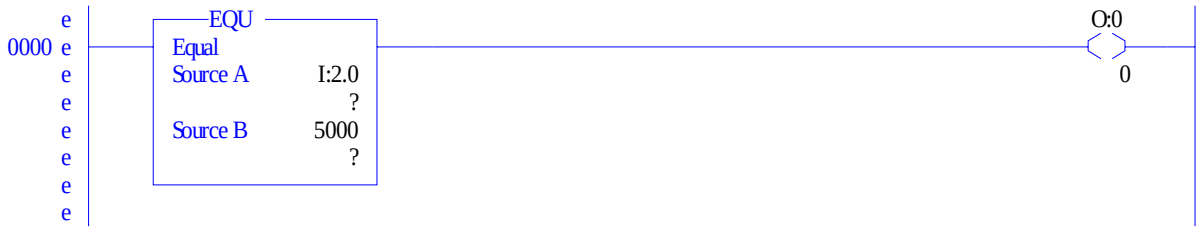
U **LIM** prozoru mogu se izmjeniti 3 polja: **Low Lim**, **Test** i **High Lim**. **LIM** naredba radi tako da uspoređuje **Test** polje s **Low Lim** (donja granica) i **High Lim** (gornja granica). (primjer kaže: Ukoliko se vrijednost analognog ulaza (**I:2.0**) nađe između **5 000** i **10 000**, uključit će se digitalni izlaz **0** (**O:0.0/0**)).

Ispitivanje točnosti izraza: $Low\ Lim < Test < High\ Lim$.

6.4.3.7 MEQ naredba (testiranje bita)

U prozoru **MEQ** naredbe mogu se izmjeniti 3 polja: **Source**, **Mask** i **Compare**. **MEQ** naredba je ustvari ispitivanje vrijednosti određenih bitova unutar **Source**-a. **Mask** određuje koji će se bitovi u **Source**-u testirati, a **Compare** je vrijednost koju oni moraju iznositi. Može se testirati najviše 16 bitova.
U ovom primjeru PLC program uspoređuje da li vrijednost odabranih bitova iznosi 128.

6.4.3.8 EQU naredba (naredba jednakosti)



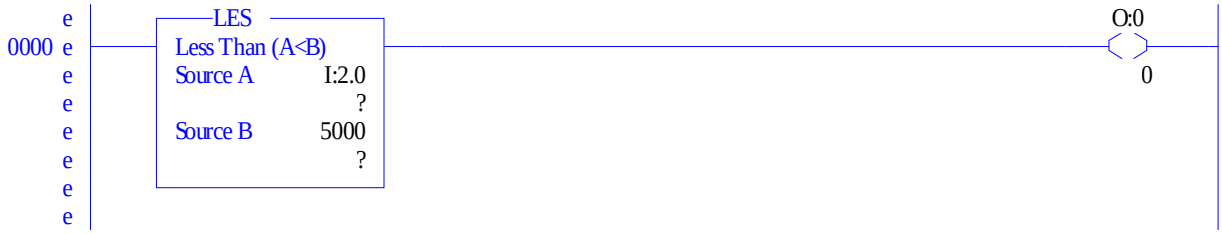
U prozoru **EQU** naredbe mogu se izmjeniti 2 polja: **Source A** i **Source B**. **EQU** naredba je naredba jednakosti. Ako su varijable **Source A** i **Source B** jednake veličine, tada će se izvršiti neka naredba na kraju rung-a (primjer kaže: Ako je **I:2.0** jednak **5 000** uključit će se digitalni izlaz **0**).
Ispitivanje točnosti izraza: $Source\ A = Source\ B$.

6.4.3.9 NEQ naredba (naredba nejednakosti)

U prozoru **NEQ** naredbe mogu se izmjeniti 2 polja: **Source A** i **Source B**. **NEQ** naredba je čista suprotnost **EQU** naredbi. Dakle, ako su varijable **Source A** i **Source B** različite vrijednosti izvršit će se naredba na kraju rung-a. (primjer kaže: Ako vrijednost **I:2.0** ne iznosi **5 000** uključit će se digitalni izlaz **0.**)

Ispitivanje točnosti izraza: $Source\ A \neq Source\ B$.

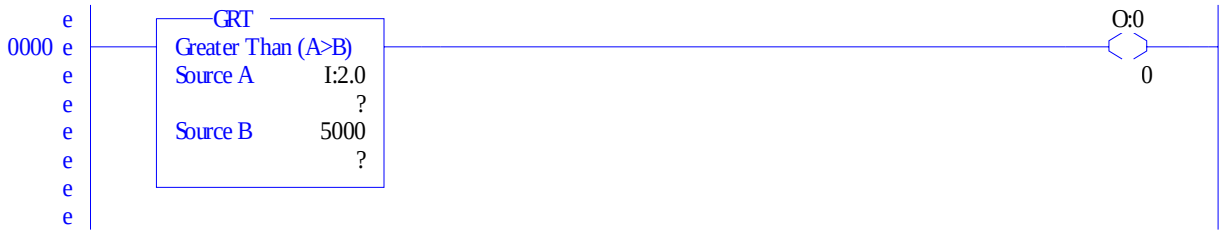
6.4.3.10 LES naredba (naredba manjeg)



U prozoru **LES** naredbe mogu se izmjeniti 2 polja: **Source A** i **Source B**. **LES** naredba uspoređuje **Source A** i **Source B** i u slučaju da je **Source A** manji izvršava se naredba na kraju rung-a. (primjer kaže: Ako je vrijednost **I:2.0** manja od **5 000** uključit će se digitalni izlaz **0**.)

Ispitivanje točnosti izraza: $Source\ A < Source\ B$.

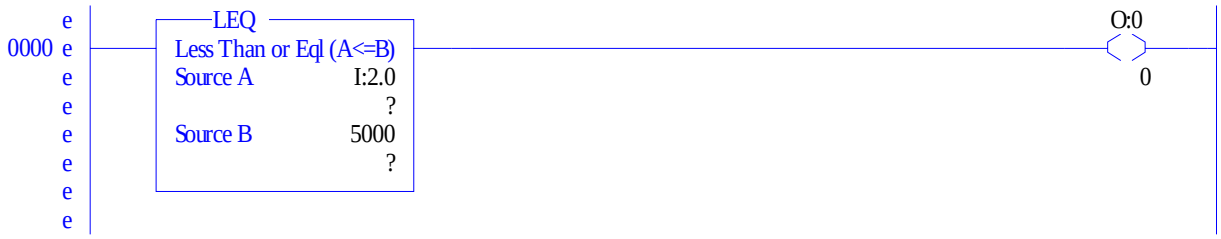
6.4.3.11 GRT naredba (naredba većeg)



U prozoru **GRT** naredbe mogu se izmjeniti 2 polja: **Source A** i **Source B**. **GRT** naredba uspoređuje **Source A** i **Source B** i u slučaju da je **Source A** veći izvršava se naredba na kraju rung-a. (primjer kaže: Ako je vrijednost **I:2.0** veća od **5 000** uključit će se digitalni izlaz **0**.)

Ispitivanje točnosti izraza: $Source\ A > Source\ B$.

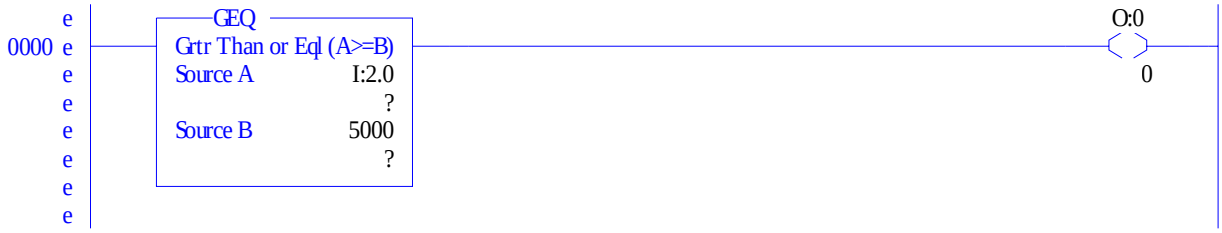
6.4.3.12 LEQ naredba (naredba manjeg ili jednakog)



U prozoru **LEQ** naredbe mogu se izmjeniti 2 polja: **Source A** i **Source B**. **LEQ** naredba je slična **LES** naredbi samo što proširuje područje provjere sa jednakošću. Znači, da bi se izvršila naredba na kraju rung-a potrebno je da **Source A** bude manji ili jednak **Source**-u **B**.(primjer kaže: Ako je **I:2.0** manji ili jednak **5 000** uključit će se digitalni izlaz **0**.)

Ispitivanje točnost izraza: $Source\ A \leq Source\ B$.

6.4.3.13 GEQ naredba (naredba većeg ili jednakog)



U prozoru **GEQ** naredbe mogu se izmjeniti 2 polja: **Source A** i **Source B**. **GEQ** naredba je slična **GRT** naredbi samo što proširuje područje provjere sa jednakošću. Znači, da bi se izvršila naredba na kraju rung-a potrebno je da **Source A** bude veći ili jednak **Source**-u **B**.(primjer kaže: Ako je **I:2.0** veći ili jednak **5 000** uključit će se digitalni izlaz **0**.)

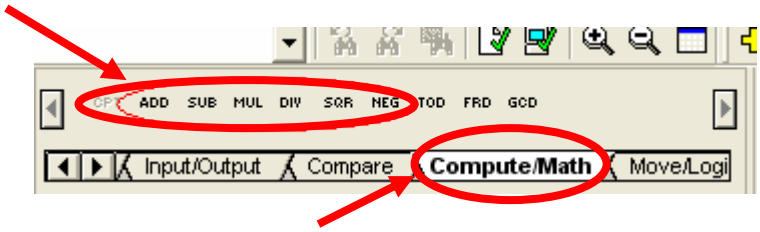
Ispitivanje točnosti izraza: $Source\ A \geq Source\ B$.

6.4.4 Rad s matematičkim naredbama

6.4.4.1 Matematičke naredbe

PLC MICROLOGIX 1500 podržava 6 matematičkih naredbi. To su: zbrajanje (ADD), oduzimanje (SUB), množenje (MUL), dijeljenje (DIV), promjena znaka (NEG), te drugi korijen (SQR).

Ikone s matematičkim naredbama



Kartica s matematičkim naredbama

Sve matematičke naredbe postavljaju se na kraj rung-a kao operacija koja će biti izvršena bude li neki uvjet zadovoljen (npr. ako smo postavili naredbu komparacije na lijevu stranu rung-a).

Opaska: 1. Mogu se zbrajati dva ista tipa podataka i pretvarati u treći (npr. cijeli broj (N7:4) + cijeli broj (N7:1) = decimalni broj (F8:2)). Iznimka je zbrajanje binarnih s ostalima tipovima podataka (decimalni broj, cijeli broj) što je nemoguće ostvariti. Isto vrijedi i za **SUB**, **MUL** i **DIV**.

2. Analogni izlaz **ne smije** poprimiti vrijednost drugačiju od postavljenih granica odabrane ljestvice skaliranja prilikom podešavanja uređaja (npr. ±40000 nije unutar skupa [-32767, 32767]). Ostali tipovi podataka: cijeli brojevi, decimalni brojevi i binarni brojevi isto imaju svoje granice koji se kreću od 16-bitne do 32-bitne riječi.

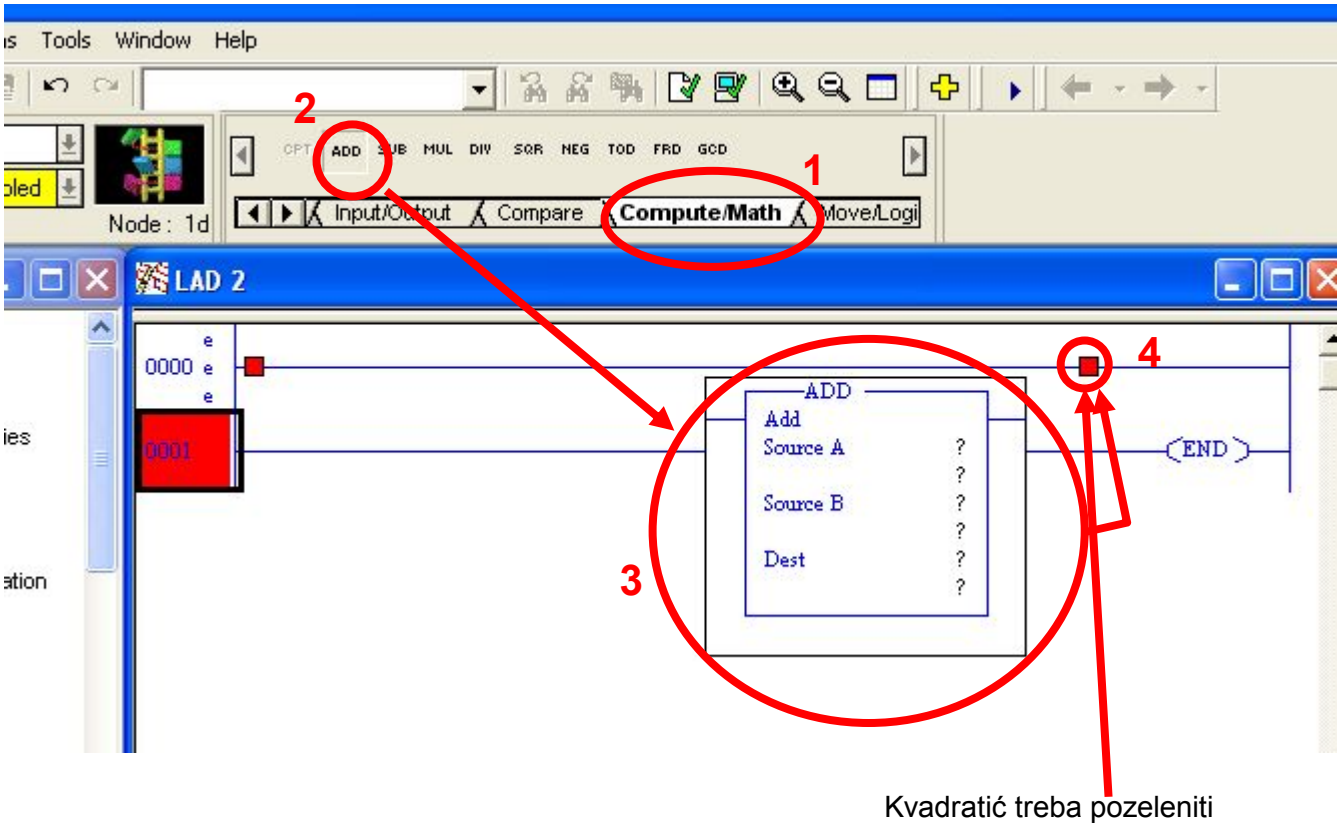
3. Ako se zbrajaju vrijednosti sa dvaju adresa, rezultat se može pohraniti u jednu od tih dvaju adresa (npr: **F:8.0** + **N:7.0** = **F:8.0**). Isto vrijedi i za matematičke naredbe **SUB**, **MUL** i **DIV**.

6.4.4.2 Postavljanje matematičke naredbe u logički krug

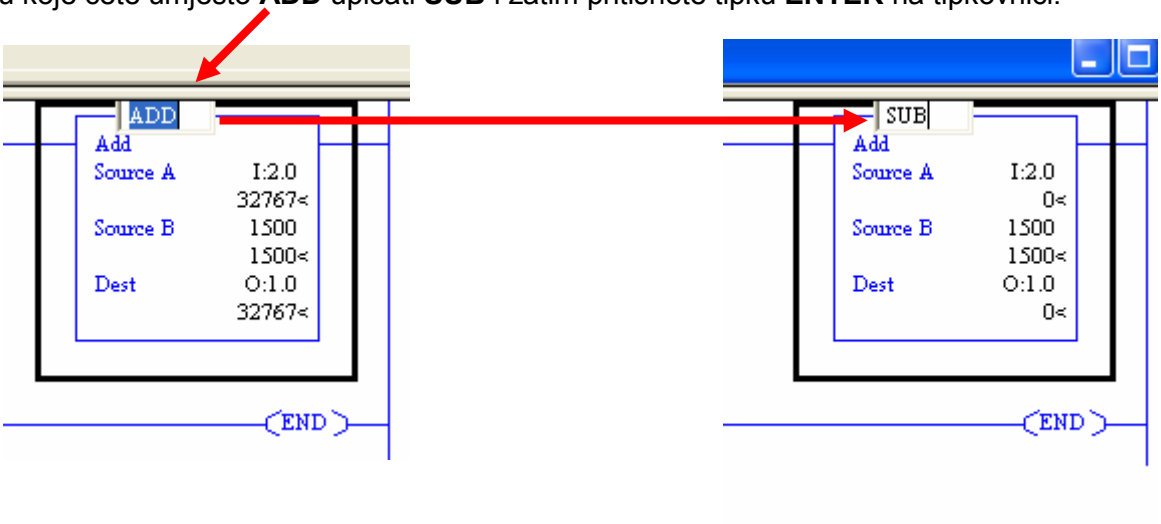
1. Dodajte slobodni rung tako što ćete prvo pritisnuti lijevu tipku miša na početak **END runga**, a zatim pritisnuti tipku **Insert** na tipkovnici.

2. Na karticama naredbi odabrat ćemo pritiskom na lijevu tipku miša karticu **Compute/Math** (**1**)

3. Pritisnut lijevu tipku miša na **ADD** naredbu i držati tipku pritisnutu (pogledati donju sliku) (**2**) sve dok je ne prenesete na kraj rung-a (**4**) (**ADD** prozorčić će se pojaviti tek kada kursor miša prijeđe na **LAD 2** prozor, a tipku miša treba pustiti tek kada kvadratić na kraju runga (**4**) pozeleni).



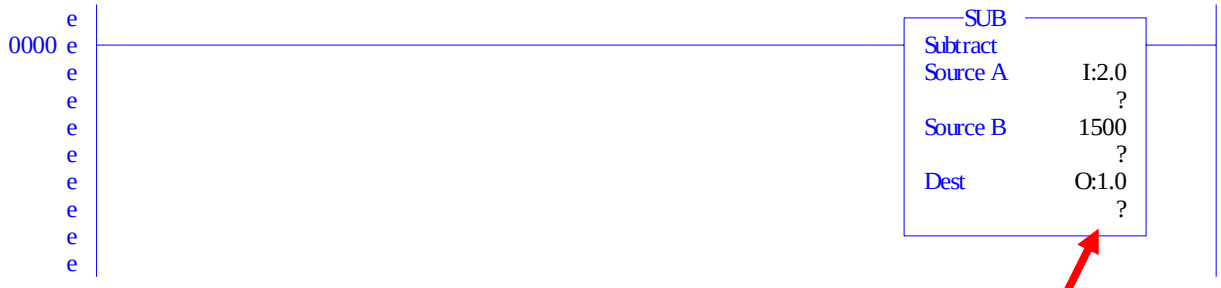
Drugi način postavljanja prozora matematičkih naredbi jest iskorištavanje postojećih. Za primjer može poslužiti **ADD** prozor koji se može «pretvoriti» u SUB prozor.
Brzim dvostrukim pritiskom na lijevu tipku miša na naslov **ADD** otvorit će se polje za upisivanje u koje ćete umjesto **ADD** upisati **SUB** i zatim pritisnete tipku **ENTER** na tipkovnici.



6.4.4.3 ADD naredba (zbrajanje)

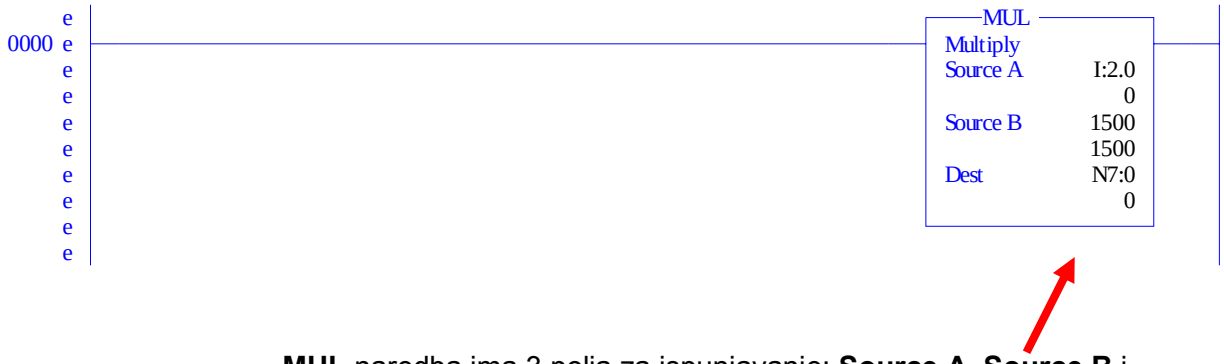
ADD naredba ima 3 polja za ispunjavanje: **Source A**, **Source B** i **Dest**.
Izraz **ADD** naredbe glasi: $Source\ A + Source\ B = Dest$. (npr. $I:2.0 + 1500 = O:1.0$)

6.4.4.4 SUB naredba (oduzimanje)



SUB naredba ima 3 polja za ispunjavanje: **Source A**, **Source B** i **Dest**.
Izraz za **SUB** naredbu: $Source\ A - Source\ B = Dest$. (npr. $I:2.0 - 1500 = O:1.0$)

6.4.4.5 MUL naredba (množenje)



MUL naredba ima 3 polja za ispunjavanje: **Source A**, **Source B** i **Dest**.

Izraz za **MUL** naredbe: $Source\ A \times Source\ B = Dest$. (npr. $I:2.0 \times 1500 = N7:0$)

> Opaska: Pazite pri uporabi ove matematičke naredbe ako koristite analogne izlaze u **Dest** polju jer produkt lako izađe iz dozvoljenih granica memorijskog spremnika. (npr. od 1*1=1 do 181*181=32761 !!)

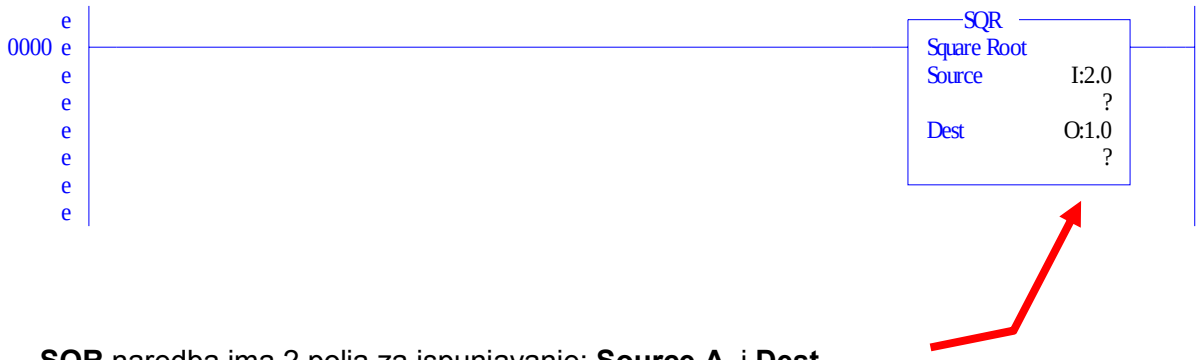
6.4.4.6 DIV naredba (dijeljenje)

DIV naredba ima 3 polja za ispunjavanje: **Source A**, **Source B** i **Dest**.

Formula za **DIV** naredbu: $Source\ A : Source\ B = Dest$. (npr. $I:2.0:3 = O:1.0$)

> Opaska: **Source B ne smije** biti 0.

6.4.4.7 SQR naredba (drugi korijen)



SQR naredba ima 2 polja za ispunjavanje: **Source A** i **Dest**.
Formula za **SQR** naredbu: $\sqrt{Source} = Dest$. (npr. $\sqrt{I:2.0} = O:1.0$)

> Opaska: **Source A ne smije** biti negativna vrijednost.

6.4.4.8 NEG naredba (promjena predznaka)

NEG naredba ima 2 polja za ispunjavanje: **Source** i **Dest**.
Formula za **NEG** naredbu: $-Source = Dest$. (npr. $-I:2.0 = O:1.0$)

6.4.4.9 SCP (Scale with Parameters) – linearna aproksimacija

Instrukcija **SCP** je izlazna instrukcija koja vrši linearnu aproksimaciju nad podacima veličine riječi prema sljedećem izrazu:

$$Y = \left(\frac{Y_1 - Y_0}{X_1 - X_0} \right) \cdot (X - X_0) + Y_0$$

X – (Input) u jednadžbi je argument od kojega računamo vrijednost gornje jednadžbe, može biti akumulator timera, analogni ulaz, ili neka memorijska lokacija

X_0 – (Input Min.) parametar funkcije koji označava minimalnu vrijednost ulaznog argumenta X

X_1 – (Input Max.) parametar funkcije koji označava maksimalnu vrijednost ulaznog argumenta X

Y_0 – (Scaled Min.) parametar koji nam govori koja će biti minimalna vrijednost funkcije Y

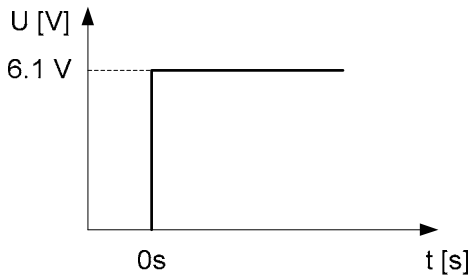
Y_1 – (Scaled Max.) parametar koji nam govori kolika će biti maksimalna vrijednost funkcije Y

Y – (Output) vrijednost funkcije koju će instrukcija **SCP** izracunati za zadani argument X, i ta vrijednost se moze slati na analogni izlaz, ili spremati na memorijsku lokaciju

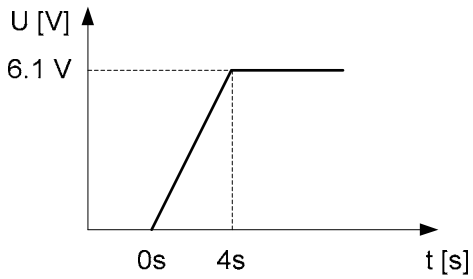
Vrijednosti sa kojima instrukcija **SCP** radi moraju biti iste velicine (16 ili 32 bita), i mogu se mijenjati za vrijeme izvođenja programa.

Primjer:

Instrukciju **SCP** možemo iskoristiti kada na analognom izlazu zelimo da napon ne skoči trenutno na neku vrijednost nego da se linearno po pravcu povećava do zadane vrijednosti.Primjer na slici:



Upotrebom instrukcije **SCP** možemo dobiti linearan porast napona na analognom izlazu do zadane vrijednosti.



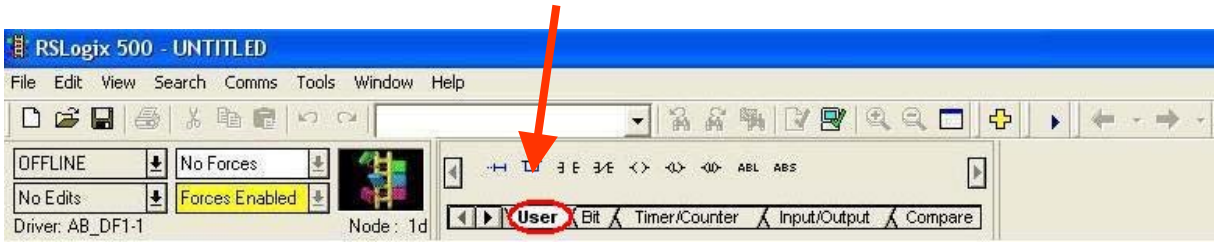
Budući da PLC radi sa digitalnim veličinama tako porast napona nemože biti kontiuiran kao na gornjem grafu nego će biti stepeničast, Ako mu za argument postavimo akumulator timera koji će brojati u vremenskoj bazi milisekunde, tada će i porast napona biti nešto manje stepeničast.

U radu sa instrukcijom **SCP** moramo paziti na sljedeće:

> Opaska: Ako za izlaz instrukcije koristimo analogni izlaz moramo paziti da nam vrijednosti funkcije Y ne budu veće od -32767 do +32767, budući da je izlaz 16 bit-ni pa ne može prihvatiti veći broj, ako dođe do toga PLC će javiti pogrešku

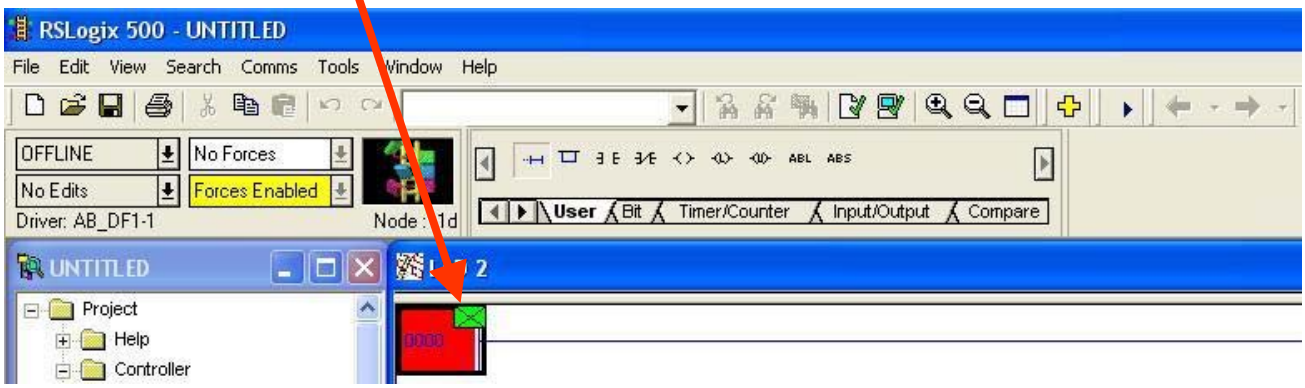
6.4.4.10 Primjer korištenja instrukcije SCP s timerom

1. Kliknemo lijevom tipkom miša na **User** karticu

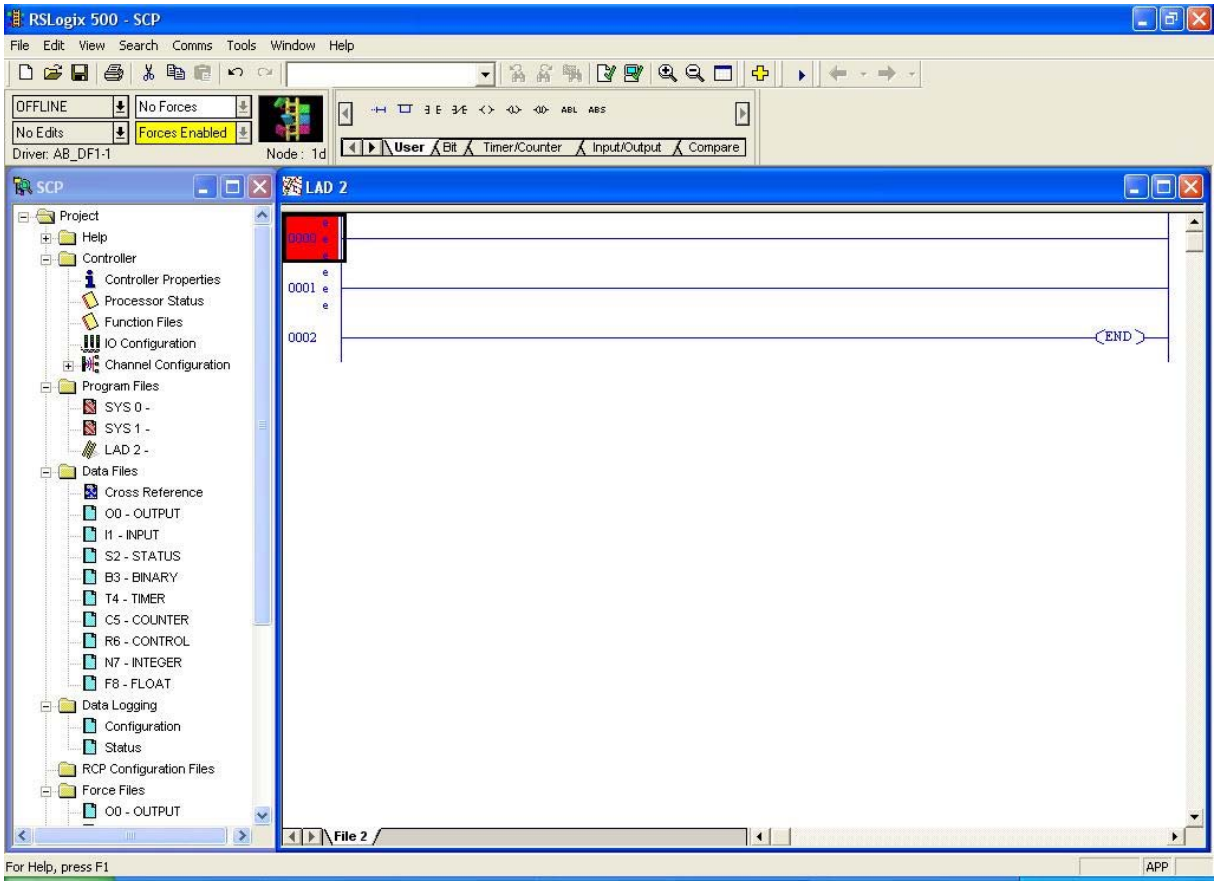


2. Na **User** kartici pronađemo **New RUNG** ikonu

3. Držeći pritisnutu lijevu tipku miša **New RUNG** ikonu dovučemo na logički krug nula sve dok se ne pojavi **zeleni X**, tada otpustimo lijevu tipku miša.

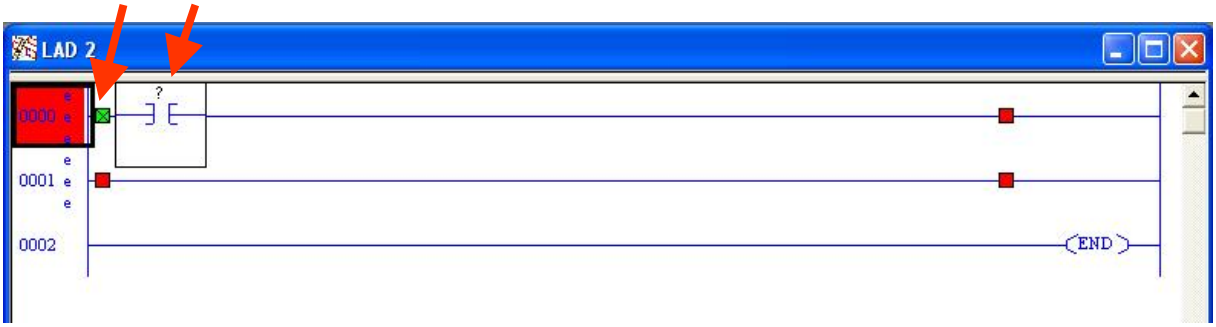


4. Sve ponovimo još jednom, sada smo dodali dva nova logička kruga, te bi program trebao izgledati kao na sljedećoj slici.



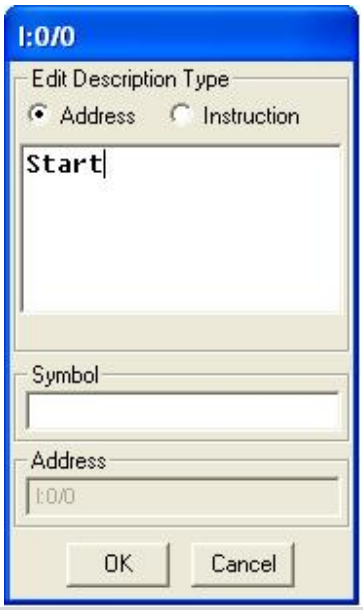
5. Iz **User** kartice odaberemo instrukciju **XIC** (Examine If Closed)

6. Držeći pritisnutu lijevu tipku miša dovućemo instrukciju do prvog logičkog kruga, kada se pojavi **zeleni X** otpustimo lijevu tipku miša, i instrukcija je dodana.



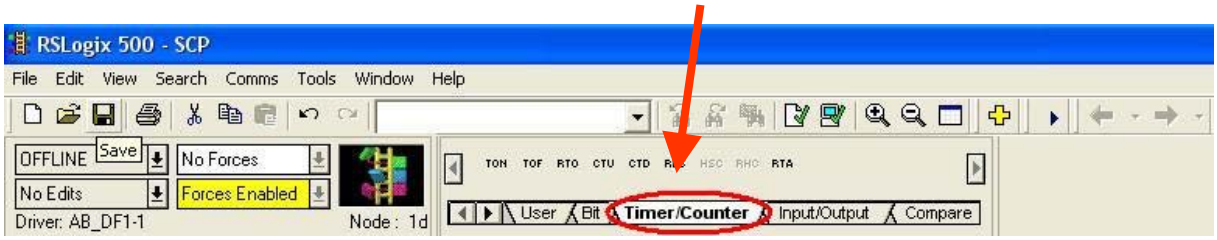
7. Zatim treba adresirati dodanu instrukciju, da bi instrukcija mogla pratiti stanje zadanog parametra (u našem slučaju nultog digitalnog ulaza `**I:0/0**`), označimo lijevom tipkom miša **XIC** instrukciju i upišemo **I:0/0**, pritisnemo **Enter** i instrukcija je adresirana.

8. Zatim će se od nas tražiti da unesemo naziv adrese. To nije neophodno, ali radi preglednosti ćemo unijeti **Start**, kliknemo na **Ok**.

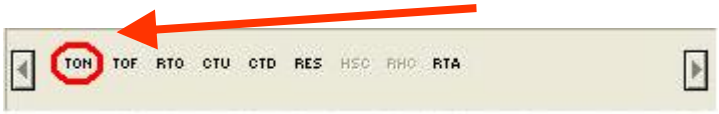


Opaska: U daljnjem radu više ne moramo baratati stvarnom adresom (**I:0/0**) nego nazivom te adrese (**Start**)

9. Kliknemo lijevom tipkom miša na karticu **Timer/Counter**



10. Iz kartice **Timer/Counter** odaberemo instrukciju **TON** (Timer On Delay)



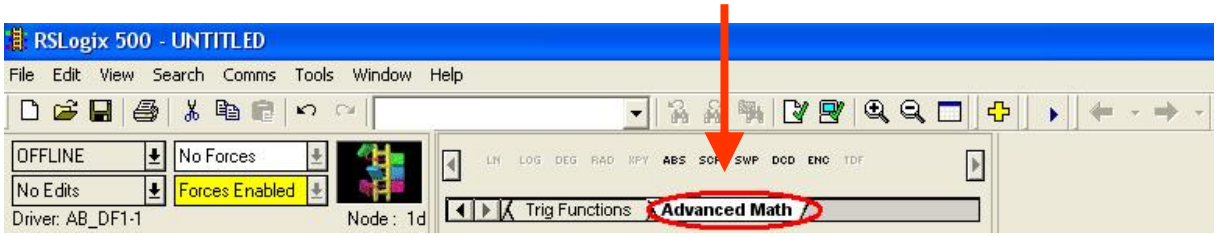
11. Držeći pritisnutu lijevu tipku miša dovučemo instrukciju na kraj prvog logičkog kruga, kada se pojavi **zeleni X** otpustimo lijevu tipku miša, i instrukcija je dodana.

12. Zatim treba podesiti parametre dodanog timera na svaki parametar dva puta kliknemo lijevom tipkom miša i unesemo željene podatke te pritisnemo **Enter**

Timer	T4: 0	adresa timera
Time Base	0.001	vremenska baza u kojoj će timer brojiti, mi odabiremo tisućinke zbog toga da dobijemo manje stepeničasti napon na izlazu
Preset	4000	vrijednost do koje će timer brojiti, kada je dosegne završit će s radom
Accum	`0`	trenutna vrijednost do koje je timer izbrojio, ta vrijednost će biti ulazni argument za instrukciju SCP

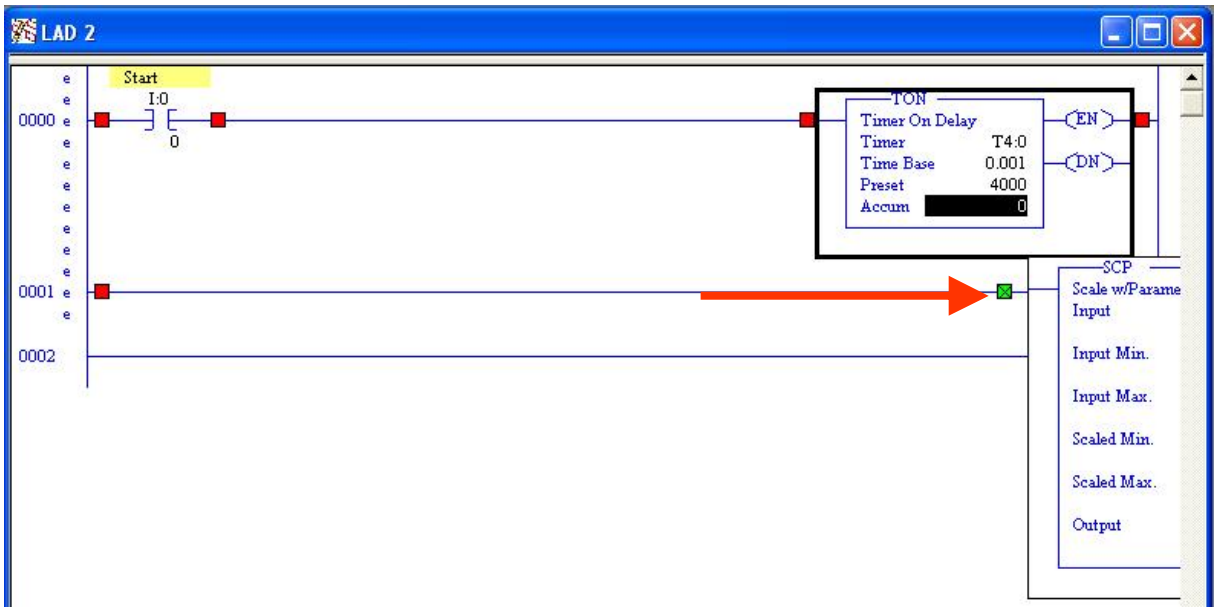
Nakon podešavanja timera program bi trebao izgledati kao na sljedećoj slici:

13. Kliknemo lijevom tipkom miša na karticu **Advanced Math**



14. Iz **Advanced Math** kartice ćemo odabrati instrukciju **SCP**

15. Držeći pritisnutu lijevu tipku miša instrukciju dovučemo na kraj drugog logičkog kruga, kada se pojavi **zeleni X** otpustimo lijevu tipku miša, i instrukcija je dodana.

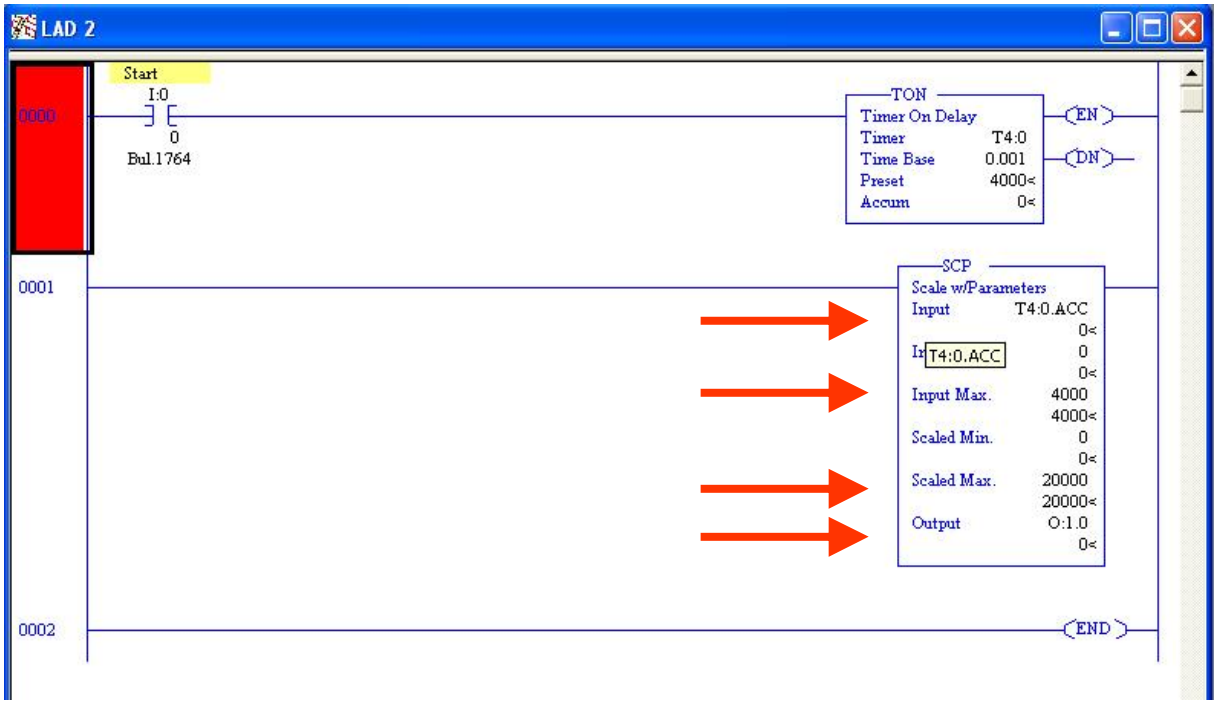


16. Zatim instrukciji **SCP** unesti odgovarajuće parametre, da bi ona obavljala zahtijevanu funkciju.

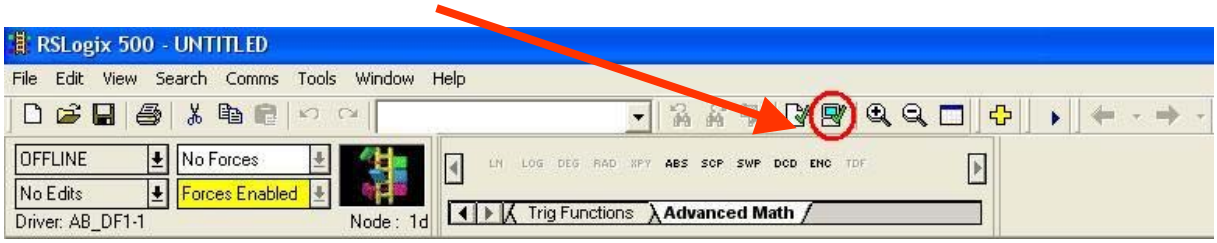
Lijevom tipkom miša kliknemo na željeni parametar i unesemo ga:

1. Input – `*T4:0.ACC*`
2. Input Min – `*0*`
3. Input Max – `*4000*` *(4 sekunde)*
4. Scaled Min – `*0*`
5. Scaled Max – `*20000*` *(6.1 VDC)*
6. Output – `*O:1.0*`

17. Nakon unosa parametara program bi trebao izgledati kao na slici ispod:

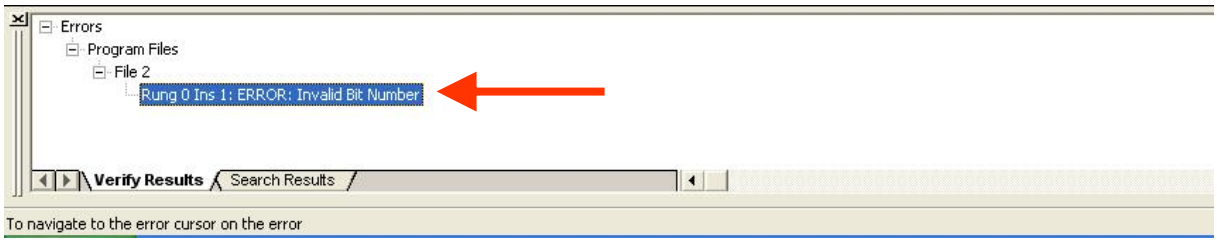


18. Slijedi provjera (verifikacija) programa, izvodi se jednostavno pritiskom lijeve tipke miša na ikonu **Verify Project**



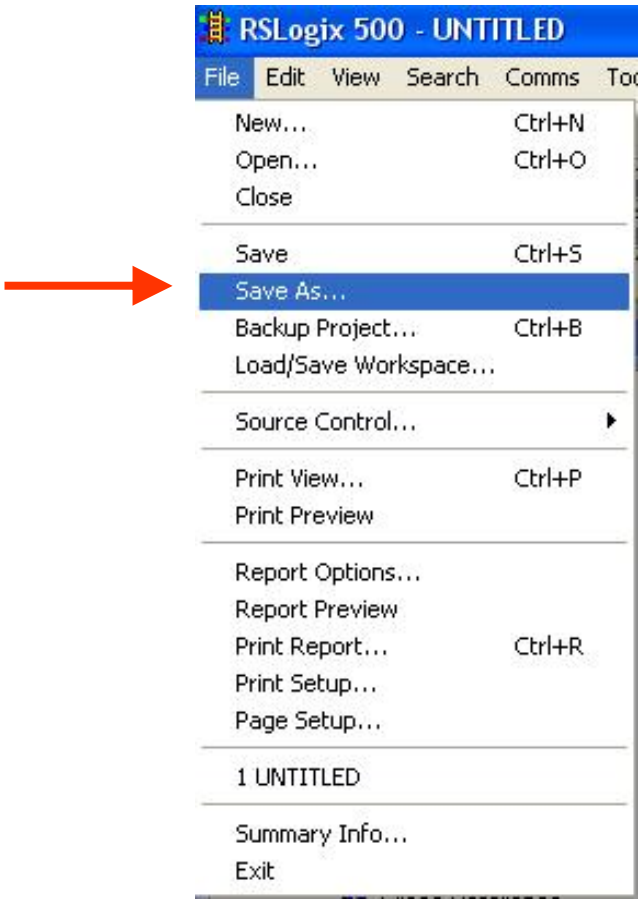
Opaska: Ukoliko provjera programa rezultira greškom, prikazat će se poruka o grešci u kojoj će biti navedeno koji logički krugovi i instrukcije sadrže grešku, pa se prema tim porukama mogu iste i ispraviti.

Na sljedećoj slici je prikazana jedna takva poruka (**Rung 0 Ins 1 : ERROR: Invalid Bit Number**):

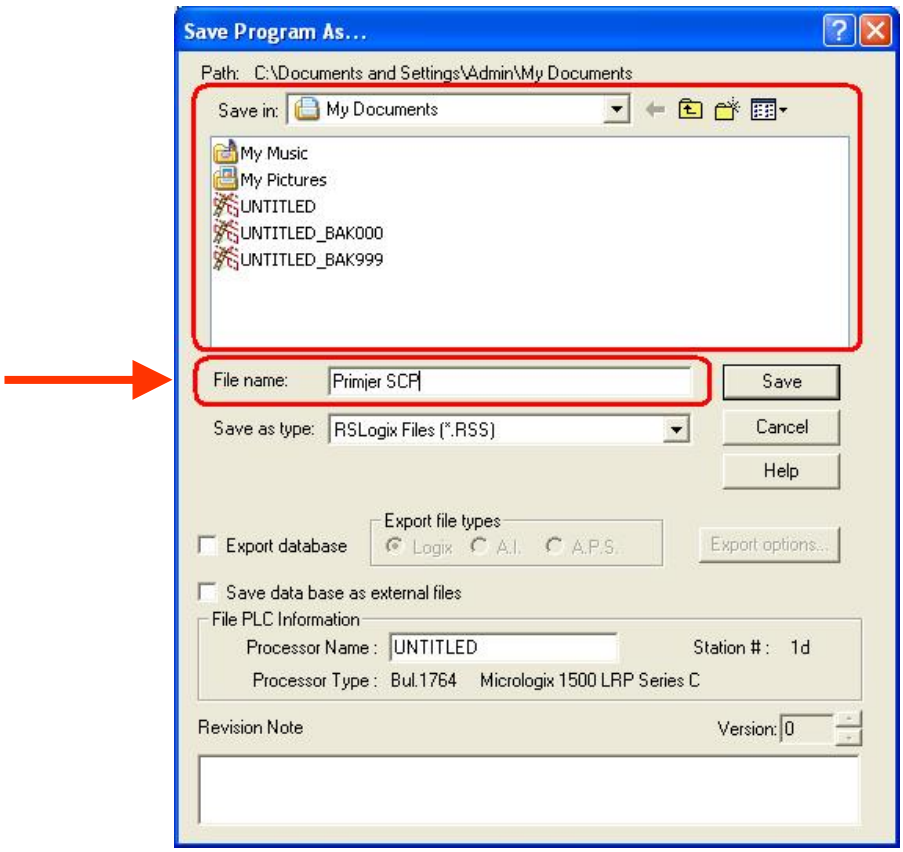


19. Nakon što su sve pogreške ispravljene i kada je program provjeren na dnu **RSLogix 500 software** ekrana pojavit će se sljedeća poruka.

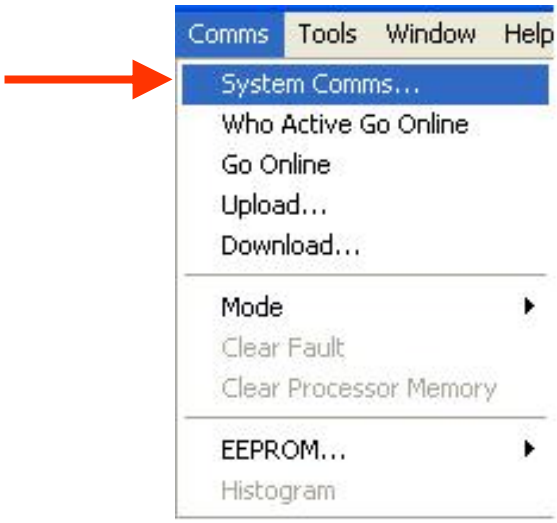
20. Nakon toga možemo pohraniti naš program, te ga downloadati na PLC. Lijevom tipkom miša kliknemo na **File** izbornik, pa potom na **Save As**



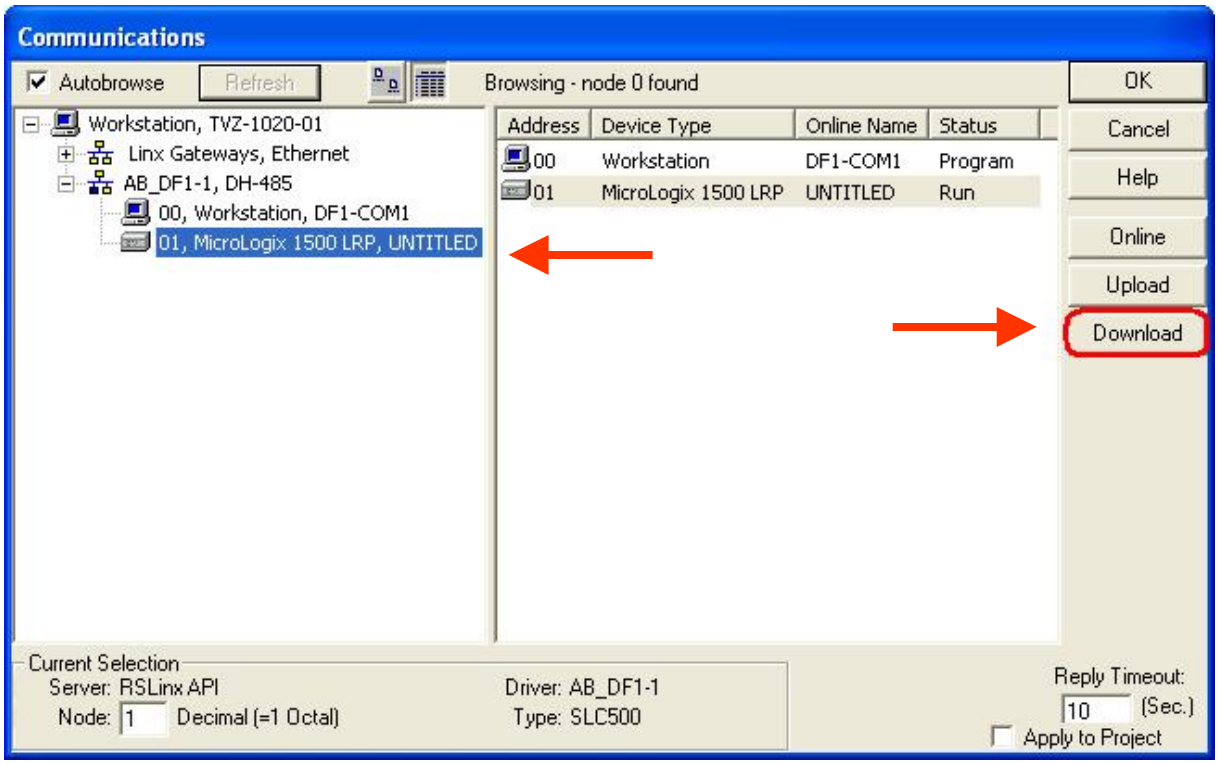
21. Nakon toga će se otvoriti prozor u kojem možemo dati ime našem programu, te odabrati u koju ćemo ga mapu spremiti. U **File name** polje upišemo željeno ime programa (u ovom primjeru to je `**Primjer SCP**`), a iz gornjeg izbornika možemo odabrati mapu u koju ćemo spremiti program.



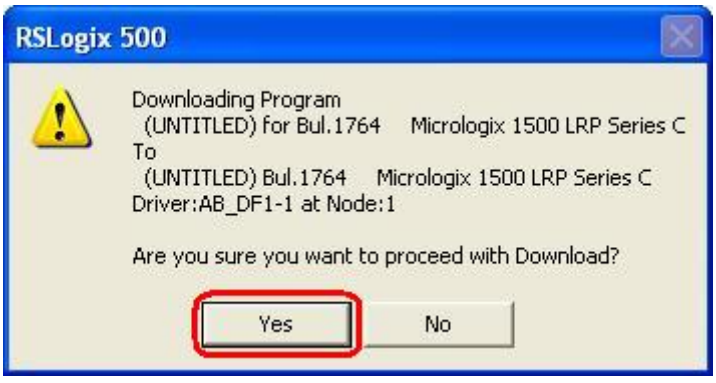
22. Nakon toga ćemo program učitati (download-ati) u memoriju PLC-a. To ćemo učiniti tako da lijevom tipkom miša pritisnemo na izbornik **Comms**, pa iz tog izbornika odaberemo **System Comms**



23. Pojavit će se prozor kao na doljnjoj slici. Ako **AB_DF-1, DH-485** driver nije otvoren, proširite driver tako da kliknete na `**+**` predznak ispred drivera.
Lijevom tipkom miša istaknemo PLC kod **Node 01**, te kliknemo na **Download**.
Ukoliko u prozoru ne postoji PLC na koji želimo učitati program, potrebno je konfigurirati **RS LINX** (vježba 1)

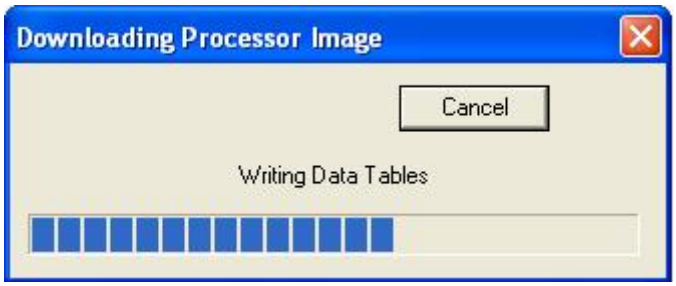


24. Nakon toga pojaviti će se prozor u kojem ćemo pritiskom lijeve tipke miša na **Yes** potvrditi download programa na PLC.



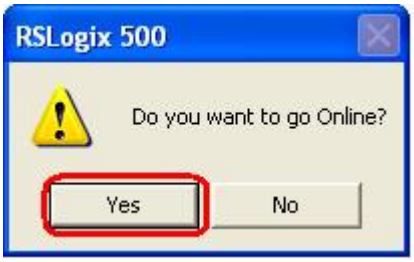
25. Pritiskom lijeve tipke miša na **Yes**, mod rada procesora se prebaci u **PROG mode** da bi mogli program downloadati u memoriju PLC-a.

26. Nakon toga će se pojaviti **Download Progress Bar**



27. Pritiskom lijeve tipke miša na **Yes** možemo procesor prebaciti u **RUN mod** da bi mogli ispitati naš program.

28. Pritiskom lijeve tipke miša na **Yes** mod rada procesora prebacujemo u **Online mod** rada.

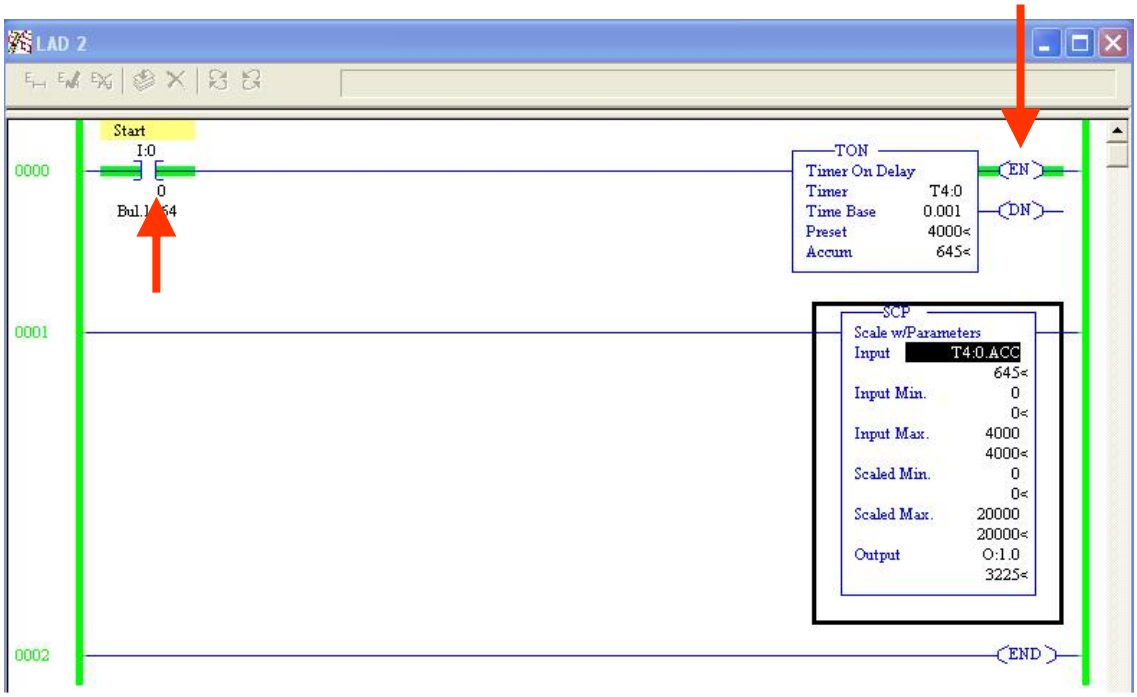


Nakon provedenog postupka program je upisan u memoriju PLC-a i pokrenut (**RUN mod**) Stanja odnosno vrijednosti pojedinih izlaza i ulaza te sam tok rada programa može se pratiti na monitoru PC-a. Uključujemo digitalni ulaz I:0/0 (**Start**) i time pokrećemo timer koji počinje uvećavati svoj akumulator za jedan svake milisekunde. Budući da smo naredbi SCP zadali akumulator timera (T4:0/ACC) za argument, tako će ona za svaku njegovu vrijednost izračunati pripadajuću izlaznu vrijednost po gore navedenom matematičkom izrazu i tu vrijednost poslati na analogni izlaz (O:1.0). Vrijednost napona će se tako gotovo linearno povećavati do vrijednosti koju postavimo za Scaled Max (u našem slučaju 20000) što pretvoreno u napon iznosi 6.1VDC. Postavljeni raspon napona za analogni izlaz je od -10V do +10V, a D/A pretvarač je 16 bit-ni što znači da rasponu napona od 0-10V odgovara 32767 različitih vrijednosti, te se izlazni napon može izračunati na sljedeći način:

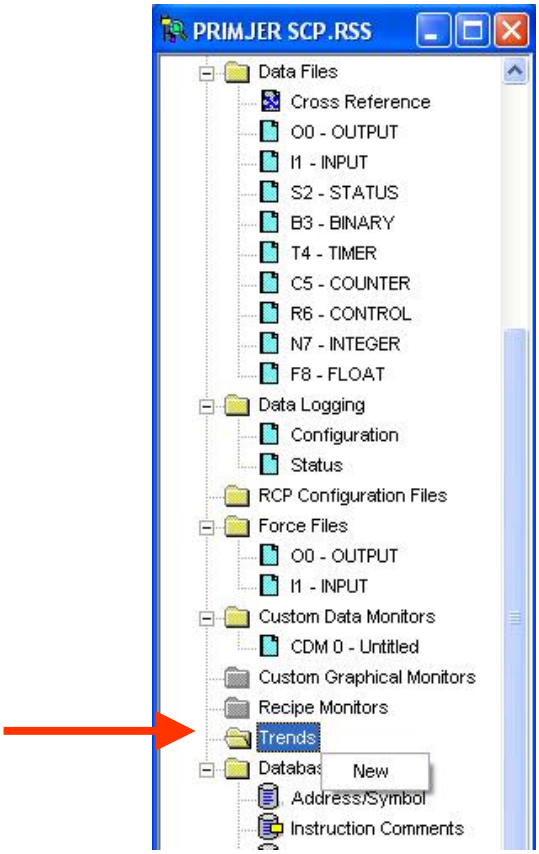
$$\frac{10}{32767} \approx 3.052 \times 10^{-4}$$

$$3.052 \times 10^{-4} * 20000 \approx 6.1V$$

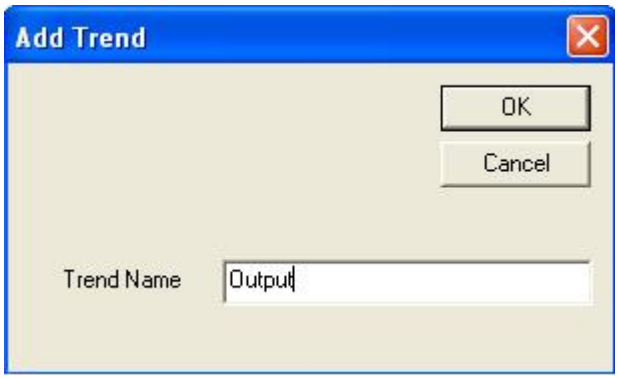
29. Kada je PLC u **Remote Run modu** rada, možemo neke njegove parametre pratiti i na računalu. Pa tako vidimo u kojem je logičkom stanju pojedini krug, vidimo stanje akumulatora timera i analognog izlaza.



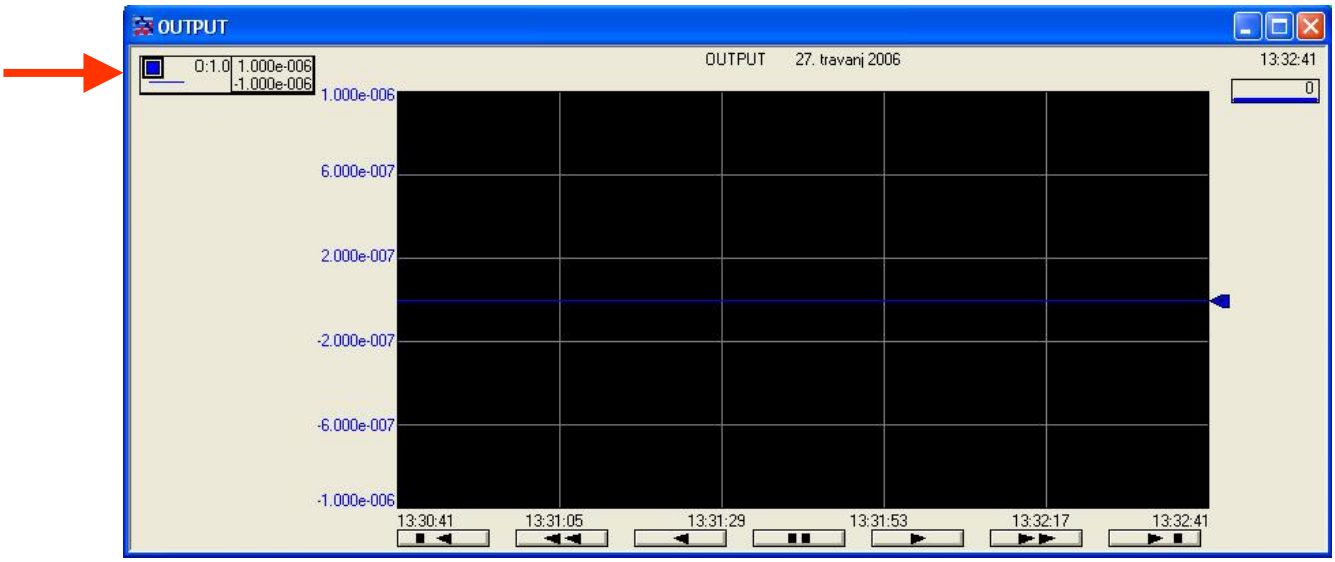
30. Isto tako možemo vidjeti kako se mijenja napon na analognom izlazu u ovisnosti o vremenu. Da bi to učinili moramo desnom tipkom miša kliknuti na karticu **Trends**, te potom lijevom odabrati **New**.



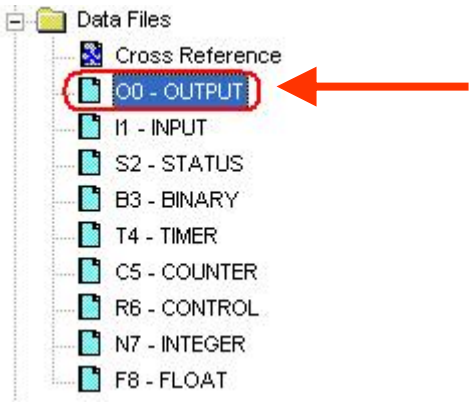
31. Moramo dati ime u našem primjeru. Nazovimo ga **Output**, te potom lijevom tipkom miša kliknemo na **Ok**



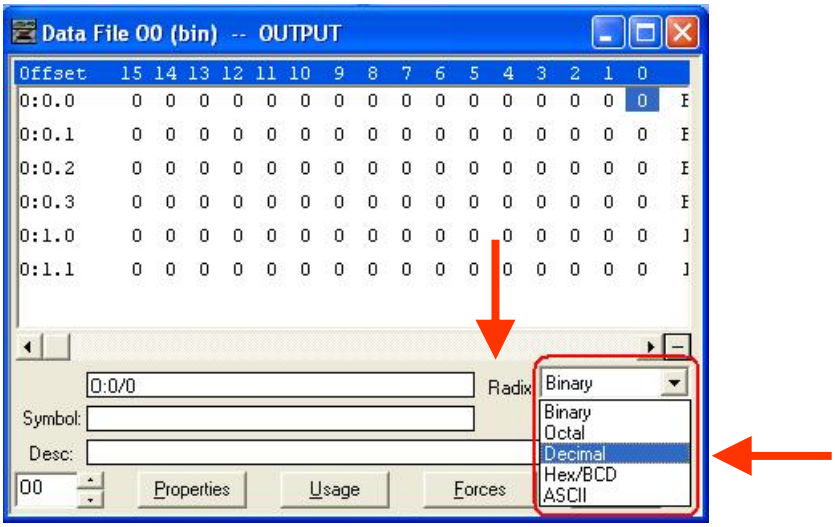
32. Dvostrukim klikom lijeve tipke miša na **OUTPUT** možemo otvoriti novo dodani trend.



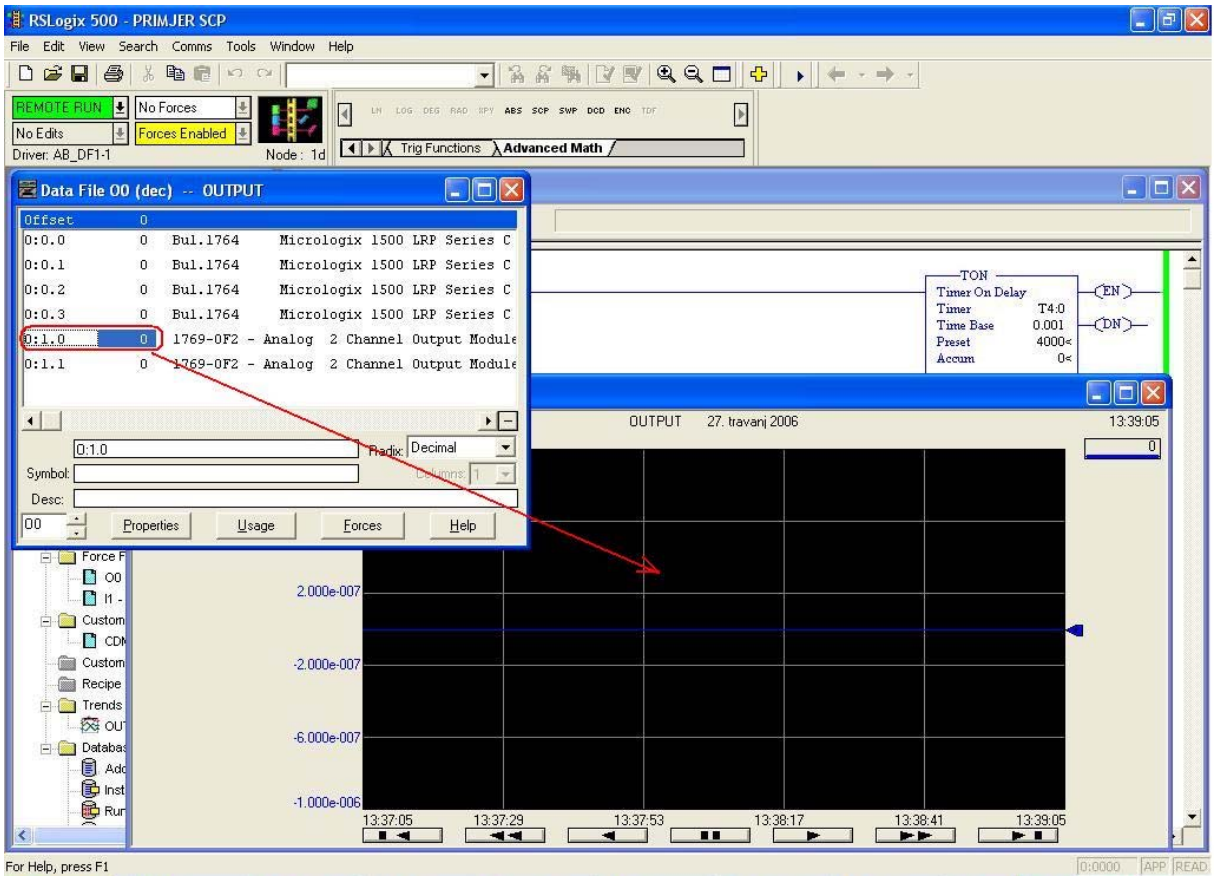
33. Moramo odabrati koji izlaz ćemo pratiti. Da bismo to učinili, dvostrukim klikom lijeve tipke miša otvorimo **Output** prozor iz kartice **Data Files**.



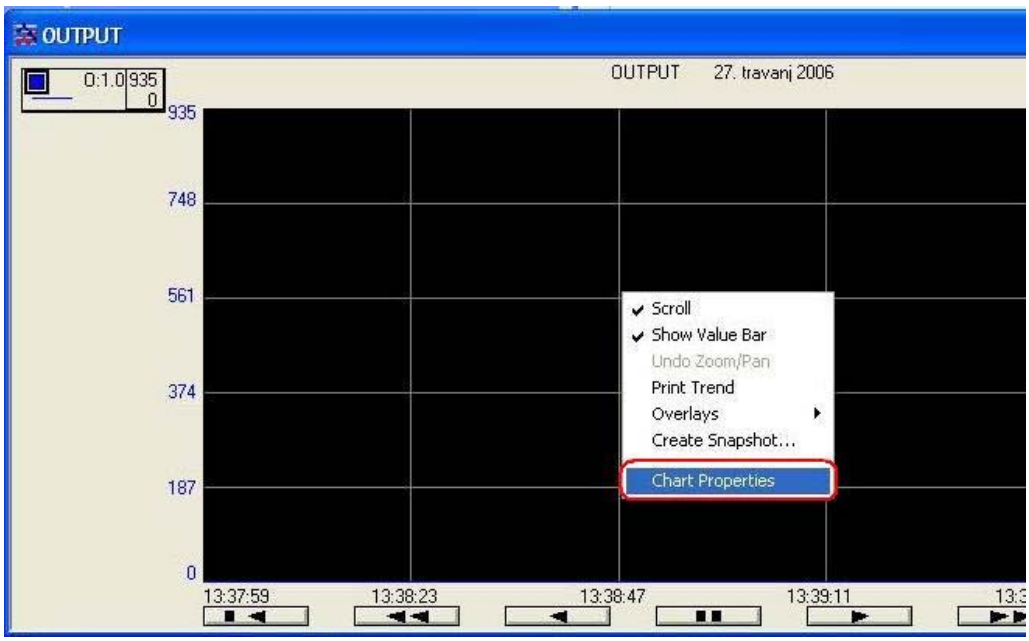
34. Iz izbornika **Radix** odabiremo vrstu prikaza. Mi ćemo odabrati **Decimal**.



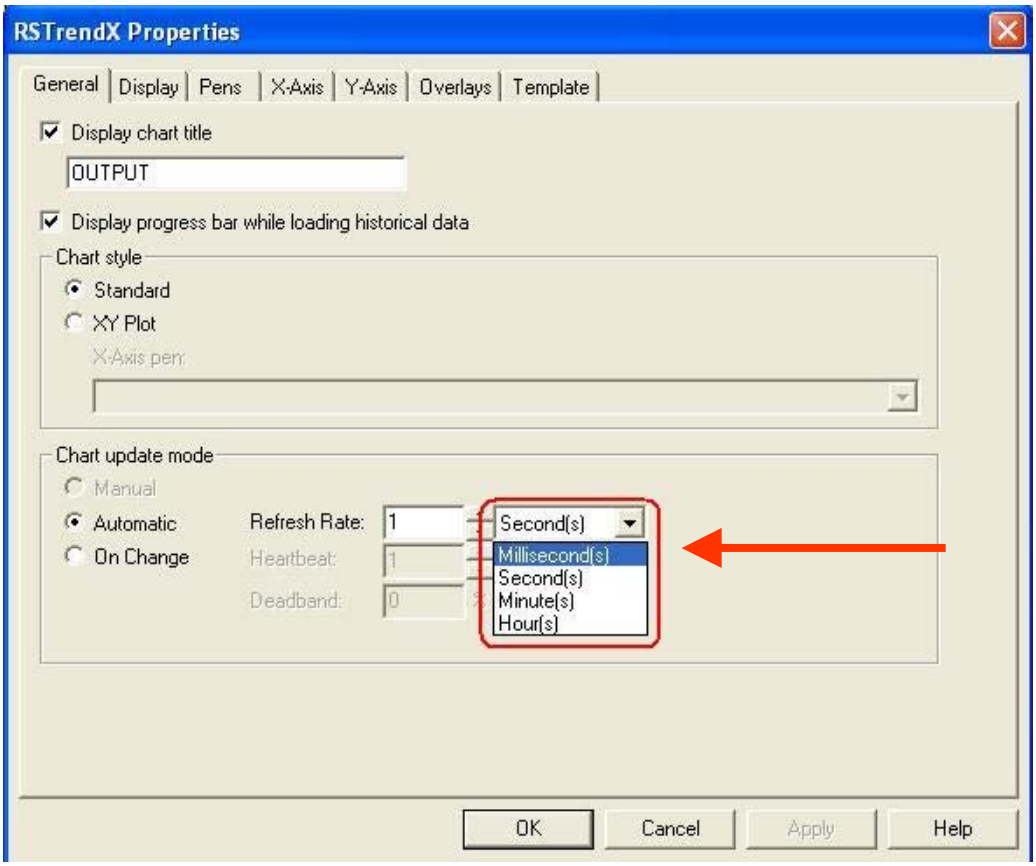
35. Prozore razmijestimo kao na slici ispod i zatim lijevom tipkom miša u **Output** prozoru kliknemo na **O:1.1** te držeći pritisnutu lijevu tipku miša vučemo u **Trend** prozor. Kada se pojavi **+** pustimo lijevu tipku miša.



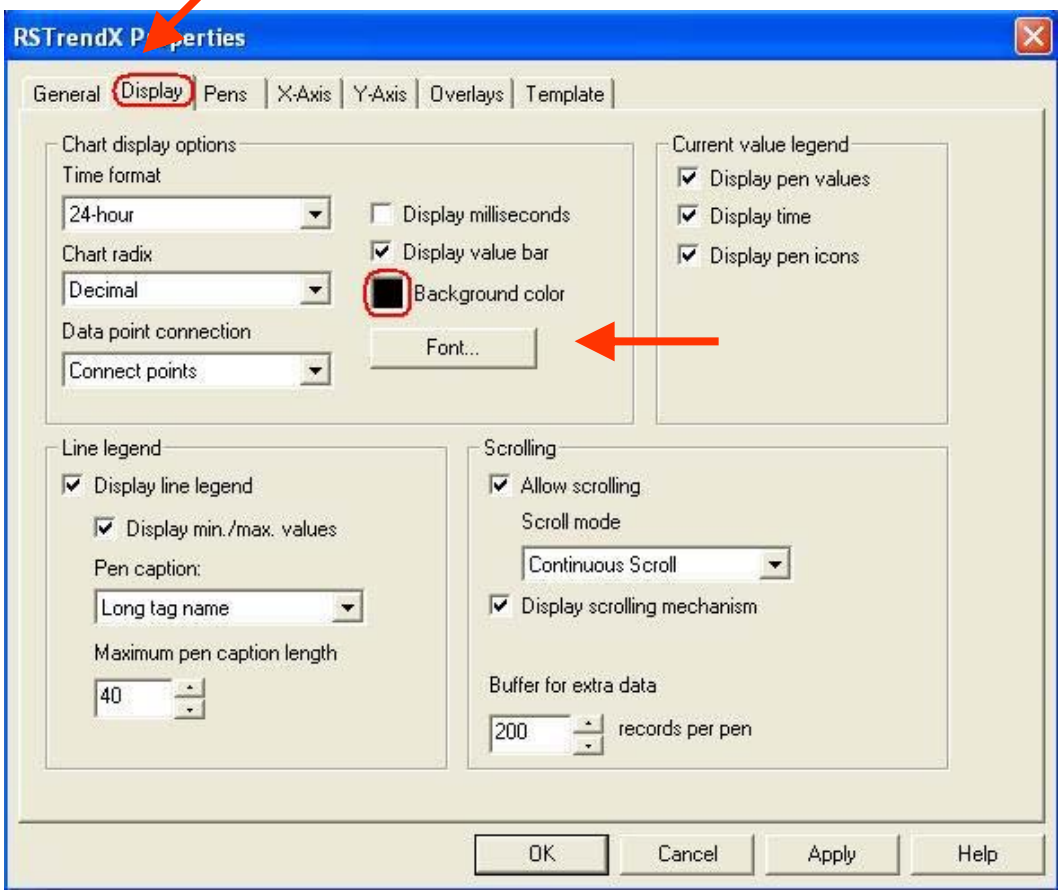
36. Ostaje nam da podesimo način na koji će RSLogix iscrtavati stanje izlaza. Desnom tipkom miša kliknemo na polje iscrtavanja grafa, te iz izbornika odaberemo **Chart Properties.**



37. Iz **Refresh Rate** izbornika odaberem milisekunde da dobijemo kontinuiraniji prikaz. Za vrijednost **Refresh Rate** upišemo 100 (100ms znači 10 očitanja u sekundi). Ljevim klikom na **Apply** potvrdimo odabir.

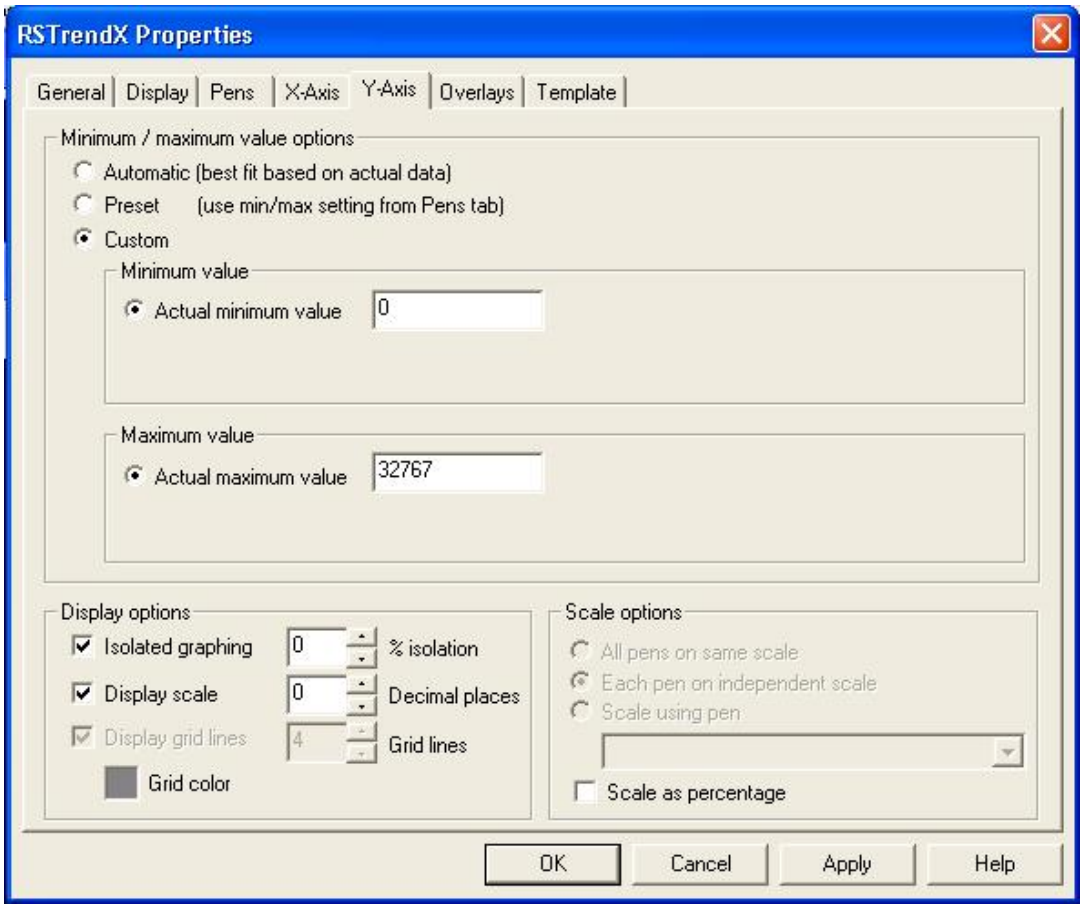


38. Iz **Display kartice** odaberemo pozadinsku boju tako da lijevom tipkom miša kliknemo na **Background color**



Možemo odabrati željenu boju, radi boljeg prikaza odabiremo bijelu i klikom lijeve tipke miša na **Ok** potvrdimo odabir.

39 Iz **Y-Axis** kartice odaberemo **Custom** i za **Actual minimum value** upišemo 0, a za **Actual maximum value** upišemo 32767. Ljevim klikom na **Apply** potvrdimo odabir.



40. Kada smo sve podesili, uključivanjem **Start** digitalnog ulaza na demo panelu pokrenuti izvršavanja programa. Trebali dobiti graf kakav je na slici ispod, iz kojeg se jasno vidi kako napon linearno raste, a potom se zadržava na maksimalnoj vrijednosti.