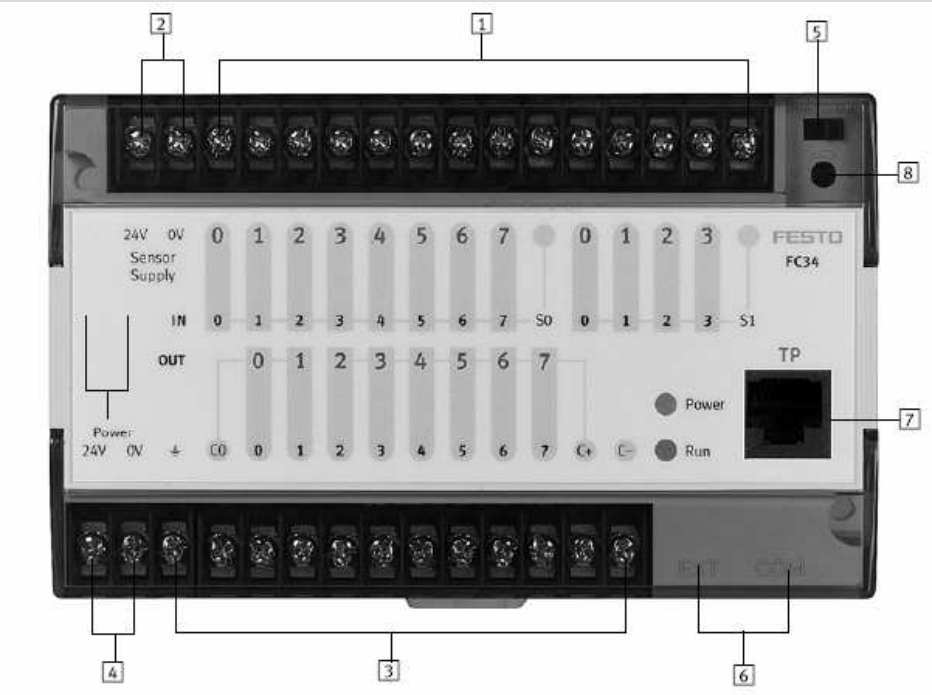
Ainārs Galiņš, Pēteris Leščevics



PROGRAMMĒJAMIE LOĢISKIE KONTROLLERI

LATVIJAS LAUKSAIMNIECĪBAS UNIVERSITĀTE
TEHNISKĀ FAKULTĀTE
Lauksaimniecības enerģētikas institūts

Ainārs Galiņš, Pēteris Leščevics

Programmējamie loģiskie kontrolleri

Mācību līdzeklis

Jelgava 2008

Mācību līdzeklis sagatavots un izdots ESF projekta „Inženierzinātņu studiju satura modernizācija Latvijas Lauksaimniecības universitātē" ietvaros, projektu līdzfinansē Eiropas Savienība.

Galiņš A., Leščevics P. **Programmējamie loģiskie kontrolleri:** mācību līdzeklis. Jelgava: LLU, 2008. – 135 lpp.

Mācību līdzeklī izklāstīta programmējamo loģisko kontrolleru uzbūve un pielietojums, aprakstītas programmēšanas valodas un programmēšanas vide, doti programmēšanas piemēri. Mācību līdzeklis paredzēts laboratorijas un praktisko darbu izstrādei lauksaimniecības enerģētikas, mašīnu projektēšanas, datorvadības un datorzinātnes, lauksaimniecības inženierzinātnes, kokapstrādes un mežistrādes inženieru specialitāšu bakalaura programmu studentiem, kuri apgūst programmējamās vadības principus un mikroprocesoru tehnikas pamatus mācību priekšmetos „Elektrotehnika un elektronika", „Mikroprocesoru vadības sistēmas", „Darbmašīnu vadības sistēmas", „Programmējamie loģiskie kontrolleri".

ISBN 978-9984-784-60-1

SATURS

IEVADS

Attīstoties elektronikai un informāciju tehnoloģijām, arvien lielāku nozīmi industrijā iegūst mikrokontrolleru vadības sistēmu izmantošana automātiskās vadības procesos. Praksē izmanto gan analogo, gan diskrēto regulēšanu. Pēdējā laikā kontrolleru izmantošana apvieno sevī gan diskrēto vadību, gan analogo vadību, analogās vadības algoritmus veidojot ar diskrētiem risinājumiem. Modernajos loģiskajos kontrolleros digitālā veidā tiek veidoti P, ID, PID regulēšanas algoritmi, pie tam tiek izmantota adaptīvā iestatīšanas sistēma. Kontrolleri tiek nodrošināti ar vairākām analogajām ieejām un izejām, kas būtiski palielina to pielietošanas iespējas. Mūsu izpratnē pazīstamais programmējamais loģiskais kontrollers PLC pēc būtības vairs nav tikai loģiskais kontrollers, bet daudzfunkcionāla programmējama ciparu vadības ierīce.

Mācību metodiskais līdzeklis „Programmējamie loģiskie kontrolleri" ir veidots, kā mācību materiāls, sākotnējās sapratnes veidošanai par loģisko kontrolleru uzbūvi, izmantošanu un programmēšanu. Kontrolleru izmantošana automātiskajās vadības sistēmās ir tehnikas virziens, kura attīstības dinamiku var salīdzināt ar datortehnikas attīstību, tāpēc izklāstītajā mācību materiālā nav likts uzsvars uz visjaunākajām attīstības tendencēm, bet uz pamatprincipiem un apmācības procesā izmantotajām iekārtām. Mācību līdzeklis ir veidots izmantojot starptautisko PLC standartu EN 61131-3 (IEC-61131-3), šis standarts paplašina programmēšanas iespējas un atvieglo lietotājiem darboties ar dažādu sistēmu un firmu kontrolleriem. Lai atvieglotu praktiskās programmēšanas apguvi, mācību līdzeklī ir doti daudzi vienkāršu automātiskās vadības procesu programmēšanas piemēri. Programmēšanai izmantotas grafiskā kāpņu diagrammu (Ladder Diagram) valoda un strukturētā teksta valoda (Statement List). Programmēšanas piemēri rakstīti un pārbaudīti ar FST programmatūru FESTO kompaktajiem programmējamajiem loģiskajiem kontrolleriem FC20 un FC34, tas neizslēdz iespēju šos algoritmus izmantot citu ražotāju kontrolleriem.

1. PLC PROCESU VADĪBAI

Jebkurai mašīnai, iekārtai, tehnoloģiskajai līnijai ir nepieciešama vadības iekārta. Atkarībā no tehnoloģijas iekārtas vadības sistēma var būt mehāniska, pneimatiska, hidrauliska, elektriska vai elektroniska. Bieži vien šīs vadības tehnoloģijas tiek apvienotas, veidojot kombinētas sistēmas. Vadības sistēma var būt cieti saprogrammēta, piemēram, vadības mehānisms vai elektroniskā vadības shēma, konkrētas operācijas izpildei. Vadības sistēma var būt viegli pārprogrammējama, piemēram, vēsturiskas iekārtas, mehānisks kontrollers ar perfokarti vai veltni, vai disku ar tapiņām, vai caurumiņiem. Nomainot perfokarti, disku vai veltni, iekārtas vadības programma tiek izmainīta.

Ja iekārtas lietotājam ir nepieciešams regulāri mainīt vadības algoritmu, ciklu skaitu vai laika iestatījumus taimeriem, tad labāk izmantot universālus kontrollerus un to elektroniskajā atmiņā ierakstīt vadības programmu. Lietotājs šo programmu var mainīt un uzlabot atbilstoši vajadzībām.

PLC – programmējamais loģiskais kontrollers (Programmable Logical Controller) ir uz mikroprocesora bāzes veidota vadības ierīce, kas aizstāj automātiskās vadības shēmas, kuras agrāk tika būvētas izmantojot relejus.

PLC ir moderna vadības ierīce, kuras pielietojums pašlaik attīstās un paplašinās. Visās nozarēs, kur pielieto automātiskās vadības sistēmas un agrāk izmantoja atsevišķus laika relejus, starprelejus un mehāniskos kontrollerus, pašreizējā mikroelektronikas, informācijas tehnoloģiju un telekomunikāciju attīstība dod iespēju tos nomainīt ar vienu ierīci, kas izpilda visas nepieciešamās vadības funkcijas.

Līdz ar PLC ražošanas apjomu pieaugumu, plašo piedāvājumu un cenu samazināšanos, gan lielās tehnoloģiskajās sistēmās, gan mazās iekārtās, arvien biežāk izmanto programmējamās vadības ierīces. Ja ražošanas procesa automātiskajai vadībai ir nepieciešami divi vai vairāki laika releji, tad lētāk un vienkāršāk ir izmantot PLC .

Šodien PLC tiek izmantots visās nozarēs, tai skaitā arī – lauksaimniecībā, enerģētikā, produktu pārstrādē, pārtikas ražošanā, metālapstrādē, kokapstrādē, autobūvē, apkures un ventilācijas sistēmās un pat „gudro māju" vadībā. PLC, kuru

vada precīzi sensori, slēdži un citi mērīšanas pārveidotāji, var ļoti precīzi ieslēgt dozēšanas iekārtas, konveijera lentas, iepakojuma mašīnas, mikroklimata iekārtas utt.

Attīstoties elektronikai un informāciju tehnoloģijām, PLC ir izveidots līdz līmenim, kas piedāvā daudz vairāk iespēju, salīdzinot ar releju vadības iekārtām. Piemēram, PLC var izmantot matemātiskās funkcijas, saskaitīšanu, dalīšanu, reizināšanu un apvienot to ar vadības funkcijām. Kontrolleros ir iestrādāti PID, ID un proporcionālie regulatori, izmantoti dažādi interfeisi, ar kuru palīdzību var nodrošināt komunikāciju ar datoru. Kontrollerus ir iespēja pieslēgt dažādiem industriālajiem un universālajiem informācijas tīkliem, LAN, CAN, I^2C. Ar kontrolleriem ir iespēja komuinicēt caur GSM tīklu, nosūtīt informatīvas un vadības komandu īsziņas un saņemt atbildes.

Eiropas ražotāji, kas izgatavo programmējamos kontrollerus (Siemens, Festo, Unitronics, ABB, Merlin Gerin, un citi) programmēšanai izmanto instrukciju valodu. Lai šo sistēmu izmantotu nepieciešamas labas iepriekšējas zināšanas un pieredze, bet priekšrocība atrodama apstāklī, ka šai gadījumā var izmantot lielu skaitu funkcijas.

Industriālās iekārtas tiek darbinātas parasti 8 un vairāk gadus, tāpēc ekspluatācijā mēdz izmantot arī iepriekšējo paaudžu kontrollerus. Pašlaik vēl ir pieejamas arī vadības sistēmas, kas var tikt programmētas, tikai izmantojot programmēšanas valodu Industrial Basic. Pēc tam, kad tehniskie darbinieki ir instalējuši vadības iekārtu, ja lietotājs pats nav kompjūteru eksperts, tam ir jāizsauc speciālisti atkārtoti katru reizi, kad rodas kaut vai neliela iekārtas darbības kļūme, neapšaubāmi, tas sadārdzina ekspluatāciju.

Mazākiem un vienkāršākiem PLC parasti ir ierobežotas instrukciju došanas iespējas. Neskatoties uz to, šo funkciju ir pietiekoši, lai varētu vadīt iekārtu ar standarta vadības procedūru komplektu, kā arī laika un matemātiskajām funkcijām.

Ja nepieciešams apstrādāt matemātisku informāciju, piemēram, kontroles displeja vērtību nolasīšanu, vai manipulatora koordinātu nolasīšanu un matemātisku apstrādi, tad jāizmanto PLC, kurš spēj to paveikt.

Lielie PLC piedāvā plašākas kodu apstrādes iespējas un instrukcijas. Šie PLC tiek galā pat ar sarežģītām programmām, piemēram, krāsu jaukšanas un tekstila krāsošanas mašīnām.

Lai varētu efektīvi izmantot kontrollerus ir nepieciešamas plašas zināšanas par automatizējamo tehnoloģiju un pieredze darbā ar ciparu vadības sistēmām.

1.1. PLC izveides iemesli

Pagājušā gadsimta 60. un 70. gados rūpniecībā sākās tendence kvalitatīvi pilnveidot ražošanu un palielināt ražošanas jaudas. Svarīgu lomu sāka spēlēt ražošanas procesu izmaiņas elastība – spēja mainīties, lai varētu sekot līdzi patērētāja vajadzībām.

60. un 70. gados automatizētas industriālās līnijas vadības aparatūra tika izvietota lielos skapjos, kas aizņēma veselas sienas. Lai varētu nodrošināt komplicētas, daudzfunkcionālas līnijas darbību, bija nepieciešams liels skaits elektromagnētisku releju. Visi releji tika mehāniski savienoti ar vadiem, to paveica izmantojot kvalificētu cilvēku resursus.

Inženieri projektēja vadības sistēmu loģiku, bet dzīvē to realizēja elektriķi mehāniski savienojot vadus. Šīs shēmas varēja saturēt vairākus simtus releju. Plāns, pēc kura tika realizēta montāžas shēma, tika saukts par *kāpņu diagrammu*. Šai shēmā ir uzrādīti visi vadības sistēmā izmantotie slēdži, devēji, dzinēji, vārsti, releji utt. Pēc šīs shēmas visus attēlotos elementus bija nepieciešams savienot.

Galvenā šādas vadības sistēmas problēma bija tā, ka iekārta būvēta uz mehāniskajiem relejiem. Mehāniskie elementi bija sistēmas vājākā vieta, jo to kustīgās daļas ir pakļautas pastāvīgam nodilumam. Ja kaut viens no relejiem pārstāja darboties, bija nepieciešams apsekot visu vadības sistēmu līdz izdevās atrast bojāto releju un to nomainīt pret jaunu.

Vēl viena šādas sistēmas problēma bija dīkstāve, kas rodas, kamēr sistēmā tika veiktas kādas izmaiņas. Ja bija nepieciešams mainīt veicamo operāciju secību, pievienot jaunas operācijas vai arī tās izdzēst, veicot pat visniecīgākās izmaiņas, sistēmu bija jāapstādina, jāatvieno no strāvas un tikai tad bija iespējams veikt šīs izmaiņas. Rezultātā ražošanas process tika apstādināts un zaudēts laiks un iespējamā peļņa.

Jāņem vērā, ka projektētājs varēja pieļaut kādu kļūdu projektējot sistēmu. Kļūda varēja rasties arī savienojot shēmas elementus, kā arī varēja pagadīties bojāta detaļa. Pārbaudīt vai sistēma darbojas varēja tikai to iedarbinot. Ja sistēma nedarbojās, bija jāvelta nogurdinošas pūles visas sistēmas pārbaudei un bojājuma atrašanai. Kamēr tika risināts defekta lokalizācijas uzdevums, viss ražošanas personāls tika pakļauts dīkstāvei, to nevarēja atļauties pat bagātākās kompānijas.

"General motors" bija pirmā kompānija, kas atzina nepieciešamību nomainīt elektromagnētisko releju vadības sistēmu. Palielinoties konkurencei automobiļu ražošanā, ražotāji bija spiesti palielināt ražošanas jaudu un kvalitāti. Nepieciešamība mainīt automatizācijas līniju vadības algoritmus, sekojot pieprasījumam kļuva ļoti aktuāla. Galvenā ideja bija, ar vadiem savienoto shēmu vietā izmantot mikrodatorus, atstājot vadības loģiku to pašu. Milzīgās vadības pultis aprīkotu ar mikrodatoru un nepieciešamās izmaiņas sistēmas darbības loģikā vai vadības operācijās, tad varētu veikt izmainot datora programmu un nevajadzēt darboties ar relejiem un savienojošajiem elementiem.

Kad tas tika izdarīts, radās jautājums - kā cilvēki, kas bija darbojušies ar relejiem tiks galā ar jauno sistēmu, kas ir pietiekoši sarežģīta un tai ir nepieciešamas programmēšanas iemaņas. Tā kā nebija iespējams montāžas elektriķus pārveidot par sertificētiem programmētājiem, "General motors" kompānija izstrādāja pirmos PLC kritērijus.

Jaunās specifikācijas bija vērstas uz iekārtas radīšanu:

- kas būtu balstīta uz elektroniskiem elementiem, mehānisko vietā;
- kurai būtu datora elastība;
- kura varētu darboties industriālā vidē, kur iedarbojas vibrācijas, augsta temperatūra, putekļi utt.;
- kuru varētu pārprogrammēt citu uzdevumu veikšanai;
- kā arī pats svarīgākais - jauno iekārtu varētu programmēt un apkalpot personāls, kas iepriekš veica releju shēmu montāžu un apkalpošanu.

Kad specifikācija bija gatava "General motors" sāka meklēt kompāniju, kas varētu izgatavot jauno iekārtu.

«Gould Modicon» radīja pirmo iekārtu, kas apmierināja izstrādāto specifikāciju. Galvenais panākums bija tas, ka iekārtas programmēšanai nevajadzēja apgūt jaunu programmēšanas valodu, bet varēja izmantot elektriķiem un tehniķiem pazīstamo *kāpņu diagrammu*. Tas kalpoja par vērā ņemamu dzinējspēku PLC attīstībā.

PLC sākumā nosauca par PC (programmable controllers). Kad parādījās PC Personal Computers programmējamos kontrolerus pārdēvēja par programmable logic controllers PLC. Sākumā PLC bija vienkāršas iekārtas, kas savienoja slēdžus un devējus ar izpildiekārtām, bet tas īpaši neatbilda sarežģītākām automatizācijas vajadzībām - temperatūras, stāvokļa, spiediena u.c. parametru regulēšanai. Ražotāji laika gaitā pilnveidoja PLC un pievienoja dažādas iespējas. Šodien PLC tiek galā ar ļoti grūtiem uzdevumiem un dažādām sarežģītām instrukcijām. Arī to ātrums un programmēšanas iespējas ir uzlabotas, kā arī ir iespējams PLC saslēgt tīklā un risināt dažādus komplicētus uzdevumus.

1.2. Darbības īpatnības

Tātad PLC nozīmē - programmējamais loģiskais kontrollers. Termins PLC tiek izmantots, lai apzīmētu vadības iekārtu, kas saskaņā ar uzdoto programmu, atkarībā no ieejas informācijas, vada procesu vai mašīnu.

Ieejas datiem ir jābūt binārajā formā, jeb strāva ir vai strāva nav, slēdzis - ieslēgts vai izslēgts - loģiskais 1 vai 0. Dati tiek apstrādāti saskaņā ar loģiskajām funkcijām, piemēram, UN, VAI, NE (AND, OR, NOT). Ieeju loģisko kombināciju, kas aktivizē izejas signālu, sauc par *vadības loģiku*. Vairākas loģiskās kombinācijas nosaka iekārtas darbību. Šo loģisko kombināciju saturs glabājas PLC atmiņā, kur tas tiek ievadīts izmantojot programmatoru - speciālu iekārtu programmēšanai, vai datoru.

Datu apstrāde notiek centrālajā skaitļošanas iekārtā - procesorā (CPU Central Processor Unit). Procesors sastāv no viena vai vairākiem mikro kontrolleriem un citām integrālām shēmām, kas izpilda vadības algoritmu un PLC atmiņas funkcijas. Procesors nolasa ieejas signālus, izpilda vadības operācijas saskaņā ar sastādīto programmu, veic skaitļošanas operācijas un padod vajadzīgo signālu uz izejām.

Ieejas signālu pārstrāde vadības komandās notiek centrālajā procesorā ar noteiktu ātrumu, kuru nosaka taktēšanas impulsu ģeneratora (Clock) darba frekvence. Vienā darba ciklā tiek nolasīti ieejas signāli, ar CPU tiek pilnībā izpildīta programma un tad tiek padoti izejas signāli. Pēc tam cikls atkārtojas.

Dažos PLC cikla laiks ir iepriekš noteikts, lai gan lielākā daļa PLC izmantojamais laiks atkarīgs no izpildāmās programmas apjoma, un tad laiku ir iespējams variēt. Vairumā gadījumu, cikla laiks tiek izteikts milisekundēs uz 1000 instrukcijām, kas vienāds ar vienu tūkstoti programmas soļu.

Vadības procesiem ar garām programmām vai daudzām, sarežģītām darbībām, vai arī mašīnām ar īsiem ieejas signāliem, garš cikla laiks var radīt problēmas.

1.3. Programmēšanas iespējas

PLC var programmēt un atkārtoti pārprogrammēt izmantojot datoru, vai speciālas programmēšanas iekārtas - pultis. Tas nozīmē, ka jebkuru PLC var programmēt ražošanas vietās, vajadzīga tikai programmatūra, bet tā parasti tiek pievienota PLC komplektācijai. PLC programmēšana ražošanas vietā ir ļoti ērta, jo rodoties kļūdām, tās turpat uz vietas ir iespējams izlabot, nepieciešamības gadījumā datorā var pārbaudīt PLC ievadīto programmu, tādā veidā pārliecinoties vai viss kārtībā. Tas dod iespēju izslēgt avārijas situāciju rašanos. Dažās ražotnēs pat tiek izmantoti komunikāciju tīkli, lai varētu regulāri pārbaudīt PLC ievadīto programmu darbības pareizību.

Programmēšana nav sarežģīta, lai apgūtu programmēšanas pamatus nav nepieciešamas liels darba un laika patēriņš. Vienkāršiem PLC programmēšanu var apgūt patstāvīgi, bet lai profesionāli ieprogrammētu lielu PLC ar milzīgu instrukciju kopumu ir nepieciešama atbilstoša izglītība un liela praktiskā pieredze.

Praksē izmanto vairākas PLC programmēšanas valodas:

- releju diagrammas, kāpņu diagrammu valoda LD (Ladder Diagram);
- funkcionālo bloku valoda FBD (AND, OR, NOT)(Function Block Diagram);
- strukturētā teksta valoda ST (Structured Text);

- secīgu funkciju diagrammu valoda SFC (Sequential Function Chart)
- programmēšanas valodas, Assambler, C, utt.

Programmējot, tiek analizēta katra kontrolējamā mašīnas darbība. Sadalot procesu vairākos posmos, tas var tikt veidots kā diagramma: katrs procesa posms (paņemšana, transportēšana, nolikšana) tiek aprakstīts atsevišķi un numurēts. Starp posmiem ir jābūt kontrolei, jo neviens posms nevar tikt uzsākts, ja iepriekšējais nav sekmīgi pabeigts. Kontroli veic gala slēdži, sensori u.c. Soli pa solim datus var ievadīt arī PLC, ja programmē ar funkciju diagrammām.

1.4. Kontrolleru konstruktīvais izpildījums

Patlaban, bez programmēšanai nepieciešamā aprīkojuma, PLC var iegādāties par pāris simtiem dolāru.

Tirgū vērojama tendence samazināties PLC cenai un sistēmas kļūst vienkāršākas. Pēc izpildījuma PLC tiek iedalīti trīs pamat grupās:

- kompaktie vienkorpusa PLC;
- kompaktie vienkorpusa PLC ar operatora paneli, jeb OPLC;
- blokveida moduļu tipa PLC.

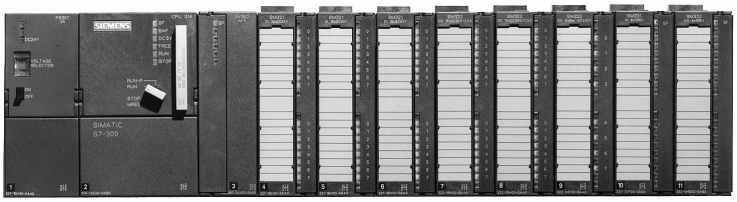
1.1. att. **Kompaktais vienkorpusa PLC FC34.**

Visas kompaktā PLC daļas - ieejas, procesors, atmiņa, barošanas avots un arī izejas, ir ievietotas vienā korpusā. Līdz ar to izmantojamais atmiņas apjoms parasti ir ierobežots, un ieejas un izejas vienību skaits strikti noteikts.

Priekšrocība vienkorpusa PLC ir zemā cena, bet trūkums - ierobežotās uzlabošanas iespējas, kā arī ierobežotā izejas jauda.

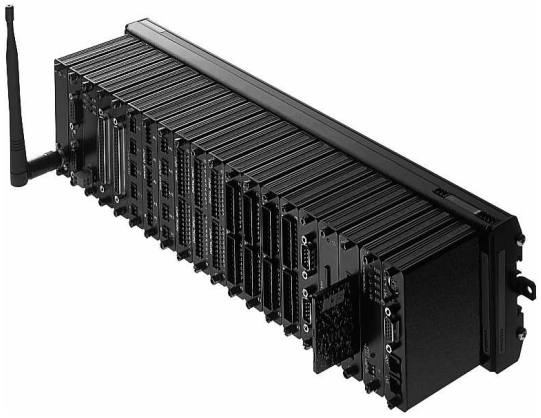
1.2. att. **Kompaktais OPLC VISION 120 ar paplašināšanas moduļiem**

OPLC ar iebūvētu grafisko displeju un operatora paneli ir ekonomiski izdevīgāki kā PLC ar atsevišķu grafisko displeju un vadības paneli. OPLC nosaukums veidots no *Graphic Operator Panel & Programmable Logic Controller*. Kontrollera galvenais bloks ir veidots, kā PLC ar noteiktu ieeju un izeju skaitu, vadība un daļēja programmēšana iespējama tieši no kontrollera vadības paneļa. Uz ierīces priekšējā paneļa izvietots grafiskais displejs procesu vizualizācijai. OPLC ieeju un izeju skaitu var palielināt ar papildus pieslēdzamiem blokiem.



1.3. att. **Moduļu tipa PLC SIEMENS S7-300**

Moduļu tipa PLC parasti sastāv no vairākiem atsevišķiem blokiem. Visi bloki tiek instalētas montāžas rāmī karšu, vai moduļu veidā - atsevišķs barošanas avots, atsevišķs procesors, atsevišķi ieejas un izejas moduļi.



1.4. att. **Industriālais kompjūters IPC PSI Professional**

Vienīgais trūkums moduļu tipa PLC ir augstā cena, bet priekšrocības ir vairākas:

- ieeju un izeju skaitu var palielināt pēc lietotāja vēlēšanās;
- var izveidot individuālu komplektāciju ar papildus blokiem;
- neierobežots skaitītāju un taimeru skaits.

Šī tipa PLC izvēle ir atkarīga no nepieciešamā minēto raksturlielumu diapazona un programmēšanas iespējām. To izdevīgi lietot lielu un sarežģītu iekārtu vadībai.

Industriālais kompjūters pēc savām veicamajām funkcijām un komplektācijas ir līdzīgs PLC (skat. 1.4. att.), bet arhitektūra un shēmas atbilst personālajam datoram. IPC sastāv no barošanas bloka, centrālā procesora moduļa, ievades izvades interfeisa moduļa, pastāvīgās atmiņas moduļa, disku nolasītāja, video kartes moduļa, bezvadu interfeisa moduļa, daudziem vadības ieeju un izeju moduļiem. Industriālie kompjūteri paredzēti darbam ražošanas apstākļos.

1.5. PLC izmantošanas priekšrocības

Tehniskajam personālam programmējamo loģisko kontrolleru izmantošana, salīdzinot ar analogajām un releju iekārtām dod vairākas priekšrocības:

- Patērētā laika ekonomija, jo pārprogrammēt var ātrāk nekā pārvienot vadus un izmainīt elektrisko shēmu.
- Elastīga iekārtas darbības izmaiņa nomainot tikai programmu un iestatījumus.
- Vienkāršāka vadības komandu došana un papildus kontrole, tāpēc, ka tādas PLC funkcijas, kā „taimers" un „skaitītājs" jau ir iebūvēti.
- Ir paveicami tādi kontroles un vadības uzdevumi, kas agrāk nebija izpildāmi - kontrole, matemātiskās funkcijas, kalkulācija, komunikācija u.c.
- Iespēja veikt iekārtas diagnostiku pieslēdzoties kontrollerim *on line* režīmā.

Turklāt pastāv arī ekonomiska rakstura priekšrocība: kontroles sistēmu, kurā jāinstalē vairāki laika releji, lētāk ir uzbūvēt, izmantojot PLC.

Ieviešot jaunākās PLC tehnoloģijas jau šodien, palielinās iekārtu konkurētspēja nākotnē.

Izdarīt pareizu izvēli starp PLC un releju vadības sistēmas ieviešanu, ir iespējams tikai tad, ja ir nepieciešamā pieredze abu šo sistēmu izmantošanā. Lai labāk iepazītu programmēšanas īpatnības, mehāniķis var uzrakstīt programmu kopā ar kontrolleru piegādātāju.

Aplūkojot attīstības iespējas citās valstīs, īpaši Vācijā, Francijā, Lielbritānijā un Zviedrijā, jāsaka, ka nākotnē mehāniķi aizvien vairāk tiks iesaistīti darbā ar PLC.

Milzīgā lietotāju pieprasījuma iemesls ir daudz elastīgākas vadības sistēmas, kā arī lielāka drošība un salīdzinoši zemās cenas.

1.6. Visbiežāk izmantotie saīsinājumi un termini

ACP (ADC)	Analogciparu pārveidotājs (analog to digital convertor). Veic analogā signāla pārveidi ciparu formā.
Adrese (Address)	Informācijas atrašanās vieta atmiņā vai ieeja/izeja uz plates.
AND	Loģiskā funkcija UN, atbilst slēdžu virknes slēgumam.
Barošanas rezervēšana (Battery back-up)	Operatīvās atmiņas aizsargāšana pret enerģijas pārtraukumu ar papildus bateriju, lai ierakstītā informācija nepazustu.
Informācijas pārraidīšanas ātrums	Pārvadīto bināro vienību skaits sekundes laikā. Ekvivalents ātrums (Bod) bitiem/sekundē.
BKD (BCD)	Bināridecimālais kods. 4-bitu kods decimālo ciparu (no 0 līdz 9) pārraidīšanai.
Bit	Datu mazākā vienība. Bita vērtība ir tikai 1 vai 0.
Būla algebra (Boolean Algebra)	Nosaukta matemātiķa Džordža Būla vārdā, kurš definējis tādas loģiskās funkcijas kā AND un NOT.
Kopne (Bus)	Datu maģistrāle, kas sastāv no paralēliem kabeļiem, pa kuriem tiek pārraidīta informācija.
Baits (Byte)	8 bitu vienība
Spole (Coil)	Termins, kas nāk no releju vadības sistēmu laikiem. To attiecina uz releja magnētisko spoli. PLC to attiecina uz 1 bita starpposmu, atmiņas izeju, izeju.

Kontrollers (Controller)	Vadības ierīce, kas pardzēta procesu vai iekārtu vadībai.
Skaitītājs (Counter)	Skaitītāja funkcionālais bloks, skaitīšanas ierīce.
Centrālais procesors (CPU)	Procesors. PLC sirds, kas dekodē un izpilda instrukcijas.
Cikla laiks (Cycle time)	Laiks, kas nepieciešams PLC, lai pabeigtu vienu ciklu: a. nolasītu ieejas datus; b. izpildītu visas komandas; c. dotus signālus uz izejām.
Dati (Data)	Informācija
Ciparu (Digital)	Ja fiziskam lielumam (strāvai vai spriegumam) var būt tikai viena no divām vērtībām (augsta (1) vai zema (0)), mums ir darīšana ar digitālu (ciparu) signālu.
Disks (Disk)	Rotējošs disks, kas klāts ar magnētisku materiālu datu glabāšanai.
Redaktors (Editor)	Programmu un tekstu izmainīšanas veids.
EPROM	Pastāvīgā, dzēšamā, programmējamā, tikai lasāmā atmiņa. Atmiņa, ko var dzēst ar ultravioletajiem stariem, pēc tam no jauna var tikt ieprogrammēta.
FIFO register (First in First out register)	Atmiņa, kurā kā pirmie ievadītie dati, kā pirmie parādās arī pie izejas. Pirmais iegāja – pirmais iznāca ārā (konveijera efekts).
Floppy Disk	Pastāvīgā atmiņa, kurai kā datu nesējs ir elastīgs 3½ collu disks, kas pārklāts ar magnētisku materiālu. Informāciju diskā ieraksta un nolasa ar speciālu ierīci, disku nolasītāju.
Galveniskā atsaistīšana (Galvanic	Ja nav pieļaujams elektriskais kontakts starp divām sistēmām, tad ir nepieciešams sistēmas galvaniski

separation)	atsaistīt. Galvanisko atsaistīšanu izmanto PLC, lai izvairītos no traucējumiem. Kā atsaistošo elementu izmanto optronu.
Aparātu nodrošinājums (Hardware)	Visas fiziskās - mehāniskās, elektroniskās u.c. aparāta vai sistēmas daļas.
Interfeiss (Interface)	Viss, kas attiecas uz PLC datu ievadi un izvadi un komunikāciju.
Instruction set	Iespējamo operāciju kopums, ko PLC var izpildīt.
Pāreja (Jump)	Pāreja (lēciens) programmā uz citu programmas punktu. Pāreja var būt: nenoteikta, tā vienmēr tiek izpildīta; noteikta, tiek izpildīta pie noteiktiem nosacījumiem.
Kbyte	Abreviatūra no Kilo Baits. 1K ir ekvivalents 2^{10} baitiem = 1024 baiti 2K ir ekvivalents 2^{11} baitiem = 2048 baiti
Kāpņu diagramma (Ladder-diagram)	Programmēšanas valoda, kurā darbība notiek it kā releju automātikas shēmā. Sauc arī par releju diagrammām.
Gaismas diode (LED)	Gaismu izstarojošs pusvadītājs, to var izmantot kā signāllampiņu.
Loģiskā funkcija (Logical function)	Darbības ar tādām binārajām vienībām, kā AND, OR un NOT.
Loģiskā shēma (Logical diagram)	Grafiska shēma, kur loģiskās funkcijas tiek attēlotas bloku veidā, kā taisnstūri.
LSB (Least Significant Bit)	Binārā skaitļa, vai vārda *jaunākās kārtas* bits.
MSB (Most	Binātā skaitļa, vai vārda *vecākās kārtas* bits.

Significant Bit)	
Atmiņa (Memory)	PLC sastāvdaļa, kur glabājas informācija.
Marķieris (Marker)	Iekšējā starpposmu atmiņa 1 bits, kurā uz laiku var saglabāt starpposmu vērtību 1 vai 0. Tās sauc par "iekšējām spolēm"
Mikroprocesors (Micro-processor)	Integrālā shēma (IC), kas izpilda datora vai PLC centrālā procesora funkciju.
NOT	Loģiskā funkcija NE, atbilst normāli saslēgtam kontaktam.
Octa	Skaitļu sistēma ar bāzes skaitli 8. Izmanto ciparus no 0 līdz 7
Off-line	Nav savienojuma (ar PLC)
On-line	Savienots (ar PLC)
Optiskais pāris (Opto-coupler)	Elektronisks elements, kas sastāv no gaismas diodes un foto tranzistora vienā korpusā - optrons. Izmanto kā galvaniski atsaistošu elementu.
OR	Loģiskā funkcija VAI, atbilst paralēlam kontaktu slēgumam.
Perifērija (Peripheral)	Aparāts PC savienošanai ar ārpasauli. Tā nav sistēmas daļa.
PLC	Programmējamais loģiskais kontrollers.
Ports (Port)	1. Loģiskā shēma ar vienu izeju un vairākām ieejām. Izejas stāvoklis pēc Būla loģikas ir atkarīgs no ieeju stāvokļa. 2. Ieejas vai izejas savienojums starp PLC un ārējām ierīcēm.
Programma	Instrukciju rinda, kas soli pa solim apraksta komandas.

RAM	Operatīvā atmiņa. Atmiņa, kas var tikt patvaļīgi ierakstīta vai patvaļīgi nolasīta. RAM parasti ir mainīga. Dati, ko satur RAM, var tikt saglabāti ar rezerves barošanas palīdzību.
Reģistrs (Register)	Vieta atmiņā, kur var būt saglabāti dati.
Relejs (Relay)	Elektromagnētisks slēdzis. Relejs sastāv no spoles ar serdi un kontaktiem. Ja spolei pieslēdz spriegumu, kontakti saslēdzas, vai atslēdzas. Ķēde, ko ieslēdz kontakti, ir galvaniski atdalīta no ķēdes, kas darbina spoli.
ROM (Read Only Memory)	Pastāvīgā atmiņa. Atmiņa, ko lietotājs var tikai nolasīt. ROM tiek ieprogrammēts jau rūpnīcā un tur saglabātā informācija nekad nepazūd.
Apakšprogramma (Routine)	Lielākas programmas neatkarīga daļa, kas izpilda specifiskas funkcijas.
RS232C (Serial interface)	Vispār pieņemts standarts virknes datu apmaiņai, protokols, kas nosaka aparātu savstarpējās sazināšanās formu (PLC sazināšanās ar printeri vai monitoru).
Skanēšanas laiks (Scan time)	Cikla laiks. Laiks, kas nepieciešams, lai vienu reizi veiktu pilnu PLC programmu.
Sequencer	Darbību secība. Operācija, kur dažādas darbības tiek izpildītas viena pēc otras.
Serial	Vārda biti tiek pārraidīti cits aiz cita pa vienu komunikāciju kanālu (virknes).
Programmnodrošinājums (Software)	Visas programmas.
SSR	Elektronisks slēdzis, kas sastāv no simistora vai tiristora. Tā ieslēgšanai nepieciešama maza strāva.

Stand-alone	Attiecas uz aparātiem, kam nevajag citas ierīces, lai tas darbotos.
Status	Stāvoklis, vērtība.
Apakšprogramma (Subroutine)	Programmas daļa ar noteiktu uzdevumu, kas var tikt izmantota vairākas reizes. Tiklīdz apakšprogramma ir izpildīta, atgriežamies pie galvenās programmas.
Sistēma (System)	Iekārtu un programmatūras kombinācija.
Taimers (Timer)	Taimers var būt gan hardware, gan software izpildījumā.
Simistors (Triac)	Elektronisks elements, pusvadītāju slēdzis, ar kura palīdzību iespējams ieslēgt un izslēgt maiņstrāvu.
VDU	Monitors.
Dežūrtaimers (Watchdog)	Shēma, kas pārbauda vai darbības tiek veiktas noteiktajā laikā. Kontrolē vai programma nav “uzkārusies”. Bieži tiek lietots, lai pārbaudītu programmas izpildes rezultātus.
Vārds (Word)	Savienotas bitu grupas, kas glabājas atmiņā.
Word-length	Bitu skaits vārdā. Parasti tie ir 8 vai 16.
Ieraksts (Write)	Uzrāda datu pārraidīšanas virzienu: no procesora uz citiem komponentiem.

2. PLC UZBŪVE UN DARBĪBAS PRINCIPS

Programmējamie loģiskie kontrolleri, neatkarīgi no izmēra, uzbūves komplicētības, izmaksām un programmēšanas iespējām sastāv no vieniem un tiem pašiem pamata komponentiem un tiem visiem ir vienādas funkcionālās īpašības. PLC vienmēr sastāv no procesora, atmiņas bloka, no ieeju un izeju blokiem un barošanas bloka. Lai nodrošinātu PLC darbību tiek izmantotas vienotas programmēšanas valodas un programmēšanas iekārtas.

PLC ir industriāla sistēma, kas sevī ietver gan aparatūras komponentes, gan programmnodrošinājumu, kas ir pieskaņots industriālai darba videi. PLC sastāvdaļu shēma redzama 2.1. att. Īpaša uzmanība tiek veltīta ieejas un izejas moduļiem, jo tiem jānodrošina procesora izolācija no apkārtējās kaitīgās industriālās vides. Kā programmēšanas iekārta parasti tiek izmantots dators un tas ar PLC tiek savienots, kad programma jāievada PLC vai kad jāmaina programmas komponentes. Kā programmēšanas valoda visbiežāk tiek izmantota kāpņu diagramma.

2.1. Pamatnostādnes

Programmējamais loģiskais kontrollers PLC, pamatnostādnes:

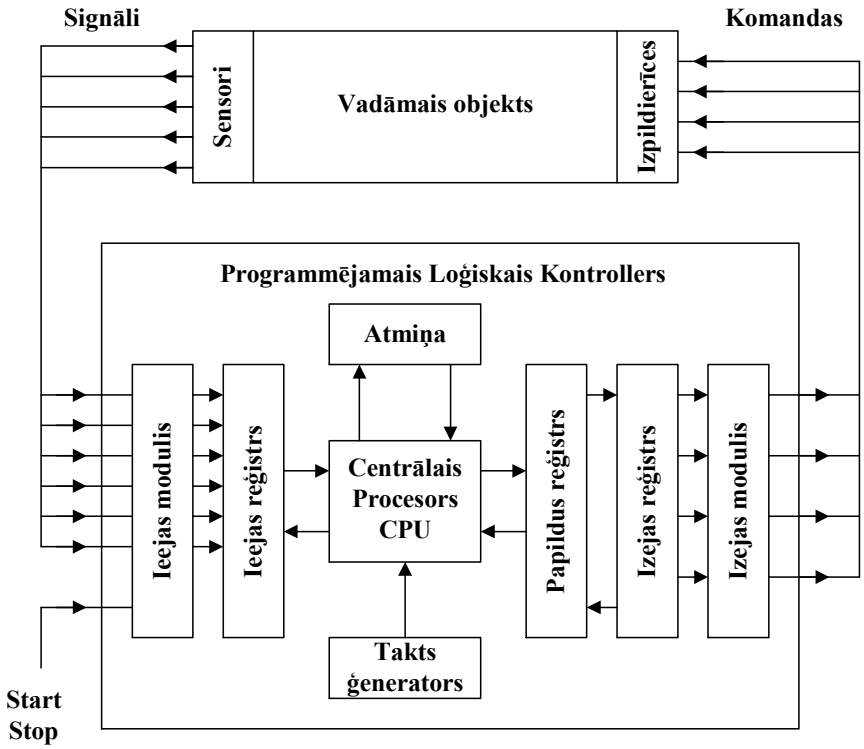
- PLC ir saīsinājums no "Programmable Logic Controller".
- PLC izpilda loģiskās funkcijas uz ievadītās programmas pamata.
- Releju ķēdes var aprakstīt saskaņā ar Būla likumu un izpildīt ar loģiskiem elementiem.
- Loģiskie elementi ir funkcijas:

  AND

  OR

  NOT
- Releju ķēdes ar loģiskajiem elementiem darbība ir atkarīga no tā, kā loģiskie komponenti ir savienoti viens ar otru.

- PLC darbība ir atkarīga no ievadītās programmas.



2.1. att. **Vadības sistēma ar PLC**

- Vadības sistēma ar PLC un PLC blokshēma dota 2.1. attēlā.
- PLC uzdevums ir apstrādāt ieejas signālus tādā veidā, lai iegūtu procesa vadībai nepieciešamos izejas signālus.
- PLC izpilda šo uzdevumu, veicot secīgu ciklisku darbību.
- PLC ir sava operāciju sistēma ar centrālo procesoru (CPU) kā kodolu.
- Procesora frekvenci nosaka taktēšanas impulsu ģenerators (clock).
- Darba cikls sastāv no soļiem.

Pirmais solis.

- Nolasa ieejas signālus („0" vai „1").
- Signālus nokopē un ievada ieejas reģistrā (atmiņā).

- Visi ieejas dati tiek izmantoti tikai vienam ciklam.
- Jauni ieejas dati tiek apstrādāti tikai pēc iepriekšējā cikla beigām.

Turpmākie soļi.

- Vadības programmas izpilde, ieprogrammētās instrukcijas tiek izpildītas cita pēc citas.
- Atkarībā no informācijas ieejas reģistrā un ieprogrammētās programmas uz izejām tiek padotas vērtības "0" vai "1".
- Izeju komandas tiek glabātas procesora papildreģistrā visu programmas izpildes laiku.

Beigu solis.

- Programma izstrādā izejas reģistra lielumus.
- Izejas reģistrs formē izejas signālu.

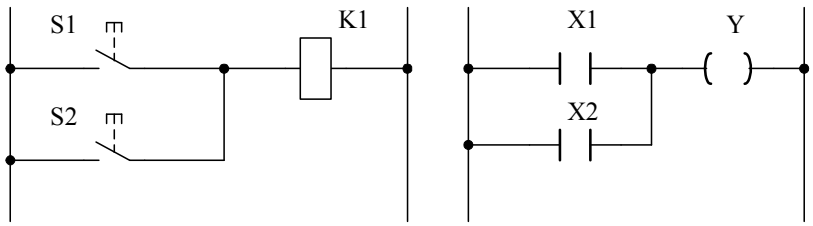
Atmiņas vietu skaits izejas reģistrā ir lielāks nekā PLC izeju maksimālais skaits.

2.2. Programmēšanas principi

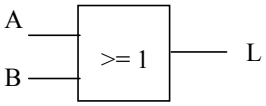
No nepieciešamā vadības algoritma analīzes var izstrādāt projekta struktūras shēmu. Projektu var rakstīt ar dažādiem paņēmieniem.

Populārākās metodes projekta struktūras attēlošanai ir:

- Principiālā shēma vai releju diagramma (skat. 2.2. att.).
- Shēma izmantojot loģiskos elementus (skat. 2.3. att.).
- Shēma izmantojot secīgo funkciju karti, SFC (skat. 2.4. att.).

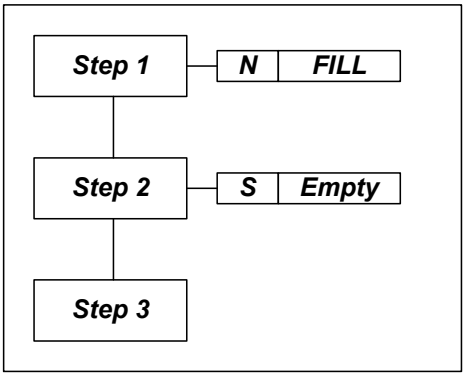


2.2. att. **Projekta struktūras attēlošana ar releju diagrammām:**
S1,S2 – slēdži; K1 – releja spole (coil); X1,X2 – kontakti; Y – spole.



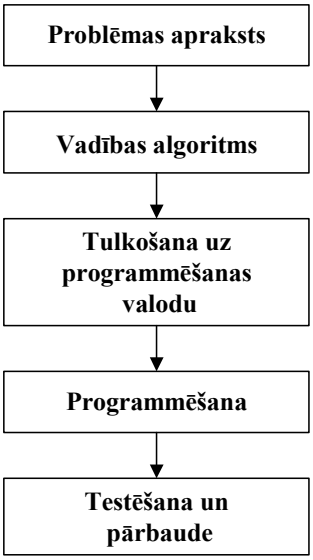
2.3. att. **Programmas attēlošana ar loģisko elementu VAI:**
A, B – ieejas; L – izeja.

Saslēguma formula: L = A + B.



2.4. att. **SFC shēmas pielietošana programmā**

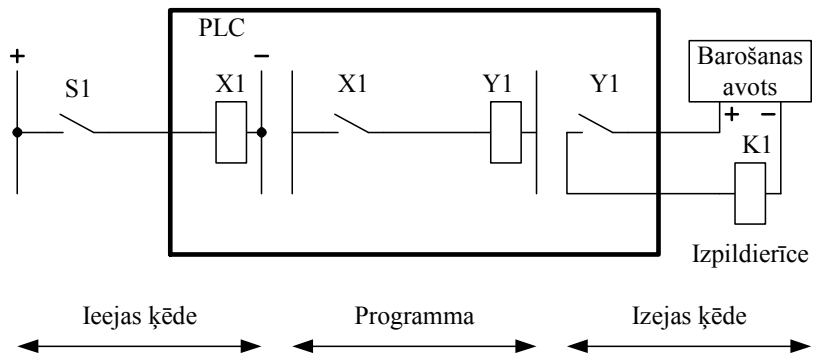
Tad vadības algoritms, kas pierakstīts, izmantojot vienu no šīm metodēm, ir "jāpārtulko" attiecīgajā PLC programmēšanas valodā.
Programmēšanas reālo uzdevumu jāizpilda tādā secībā kā parādīts 2.5. attēlā.



2.5. att. **Programmēšanas secība**

2.3. PLC darbības princips

Uzprogrammēts PLC izpilda galvenās procesa vadības funkcijas, un tas ir mijiedarbības modulis starp informatīvo (mērījumu) un izpildes (pārvades) daļām vadības sistēmā.

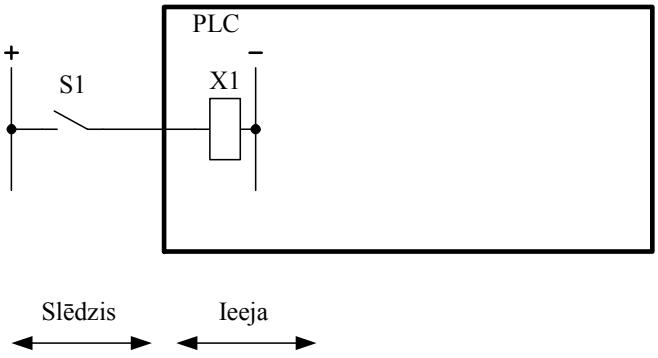


2.6. att. **PLC pielietošana vadības sistēmā**

PLC vadības sistēmas modeļa darbība (skat. 2.6. att.):

- Kontakti S1 nav saslēgti.
- Ieeja X1 nav aktivizēta, ieejas statuss ir 0.
- Ieprogrammētais kontakts X1 nav saslēgts.
- Ieprogrammētā izeja Y1 nav iedarbināta.
- Kontakts Y1 nav saslēgts, izejas statuss ir 0.
- Relejs K1 nav aktivizēts.
- Kontakti S1 saslēgti.
- Ieeja X1 aktivizēta, ieejas statuss – 1.
- Ieprogrammētais kontakts X1 saslēgts.
- Ieprogrammētā izeja Y1 aktivizēta.
- Kontakts Y1 saslēgts, izejas statuss – 1.

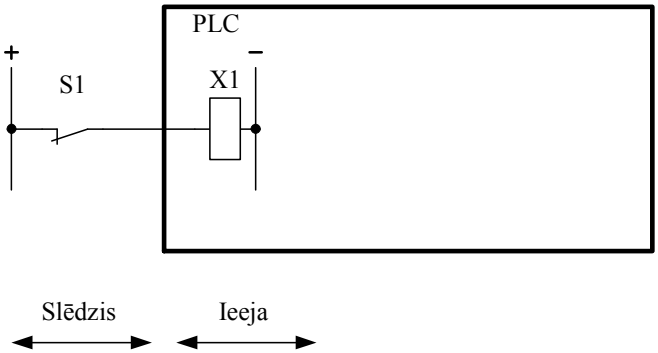
Lai izvairītos no sajukuma normāli vaļējo un normāli saslēgto kontaktu izmantošanā par ieslēgšanas elementiem, nepieciešams pārskatīt attiecīgo ieeju (2.7. un 2.8. att.) un izejas (2.9. att.) statusu.



2.7. att. **PLC ieejas vadībai normāli vaļēji kontakti**

Ieejā slēdzis ar normāli vaļējiem kontaktiem - saslēdzošais (skat. 2.7. att.):

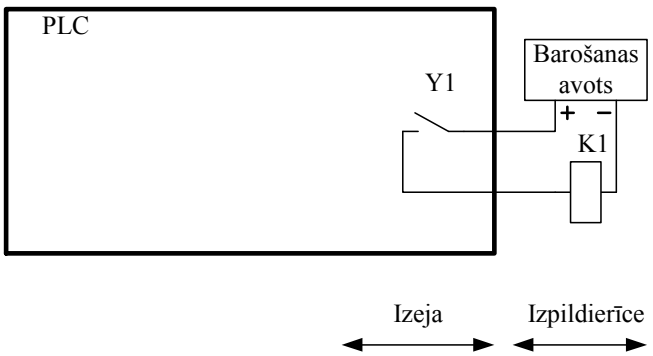
- S1 kontakti nav saslēgti, ieejas statuss – 0.
- S1 kontakti saslēgti, ieejas statuss – 1.



2.8. att. **PLC ieejas vadībai normāli slēgti kontakti**

Ieejā slēdzis ar normāli slēgtiem kontaktiem- atslēdzošais (skat. 2.8. att.):

- S1 nav nostrādājis, kontakti saslēgti, ieejas statuss – 1.
- S1 nostrādājis, kontakti atslēgti, ieejas statuss – 0.



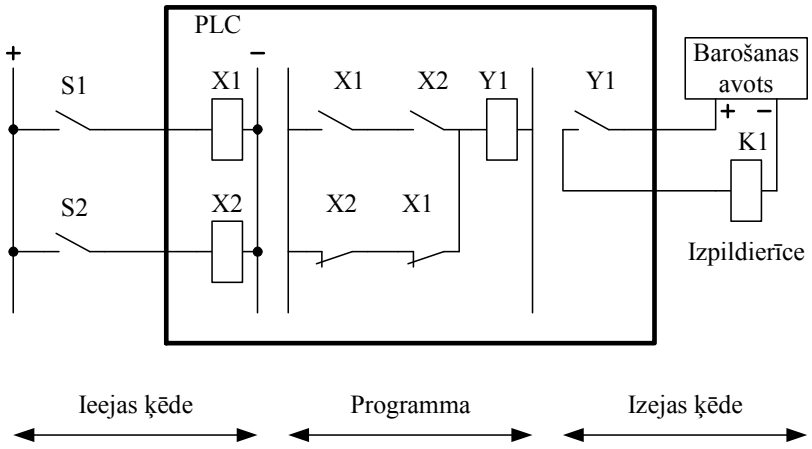
2.9. att. **PLC izeja ar normāli vaļējiem kontaktiem**

PLC izeja ar normāli vaļējiem kontaktiem (skat. 2.9. att.):

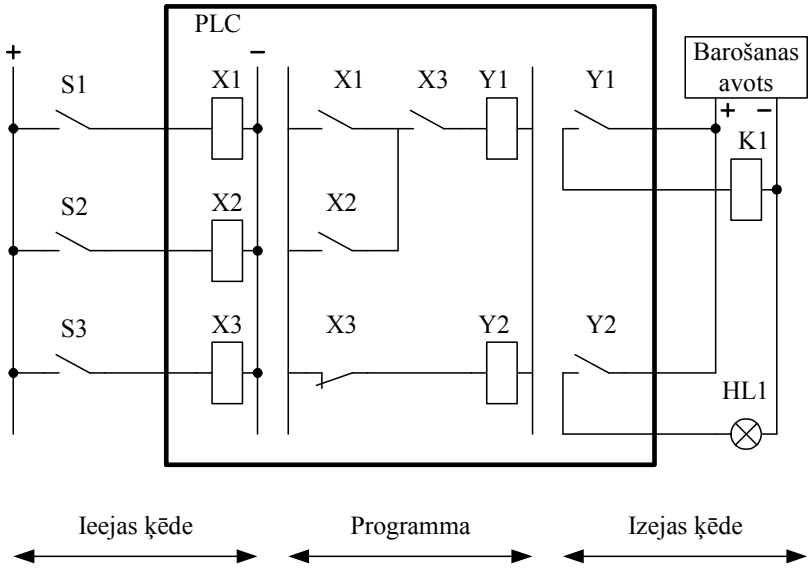
- Y1 kontakti nav saslēgti, izejas statuss – 0.
- Y1 kontakti saslēgti, izejas statuss – 1.

Piemērs

PLC tika programmēts atbilstoši shēmai, kas parādīta PLC modelī (skat. 2.10. att.). Kad tiks aktivizēta izpildierīce - relejs K1? Kādu loģisko funkciju izpilda šāds slēgums?



2.10. att. **Vadības sistēma**



2.11. att. **Vadības sistēma**

Piemērs

Vadības sistēmas modelis ir dots 2.11. attēlā. Atbildiet uz jautājumiem:

- Relejs K1 ieslēdz dzinēju.
  Kādā gadījumā tas notiks?
- Pie normāla spiediena slēdzis S3 ir saslēgts.
- Spiediens ir pārāk zems, signalizācijas lampiņai HL1 ir jādeg, bet tas nenotiek.
  Kādam jābūt Y1 statusam?
  Kādam jābūt spriegumam paralēli izejai Y1?
  Kādam jābūt spriegumam uz lampiņas spailēm?
- Visas izmērītās vērtības ir pareizas, kāpēc lampiņa nedeg?

2.4. PLC darbības detalizēti pētījumi

PLC sastāv no vairākiem atšķirīgiem komponentiem. Kompaktajā PLC visas sastāvdaļas ir ievietotas vienā korpusā. Moduļu tipa PLC šie komponenti kā moduļi ir iemontēti pamatblokā. Pamatbloks nodrošina savienojumu starp tajā esošajiem moduļiem pēc kopņu (bus) sistēmas.

PLC sastāv no šādiem, galvenajiem moduļiem:

- centrālā procesora modulis (CPU);
- ieejas modulis;
- izejas modulis;
- barošanas modulis.

Centrālais procesors (CPU)

Releju vadības sistēmai un vadībai, kas izveidota no bezkontaktu elementiem, informācija tiek apstrādāta vienlaicīgi, jeb paralēli.

Vadības sistēmā ar PLC informācija tiek apstrādāta secīgi, jeb sērijveidā, izmantojot loģiskās funkcijas. Šo procesu vada centrālā procesora taimers - taktēšanas impulsu ģenerators (clock – pulkstenis).

Informācija PLC tiek ievadīta pa pieslēgtajām ieejām.

Piemēram, ieejas ir: IX0, IX1, IX2, IX3, IX4, IX5 un IX6,
bet izejas: QX0, QX1, QX2.

Atkarībā no datiem, kas ievadīti caur ieejām un programmas, kas ieprogrammēta PLC, tiks aktivizētas atbilstošās izejas.

PLC programma sastāv no vairākām programmas rindām, kas ir saglabātas programmas atmiņā. Programmas rinda sastāv no vārda adreses un paša komandas vārda. Katrai programmas rindai ir sava adrese, sākot ar 0000, 0001 u.t.t. Katrā programmas rindā tiek ierakstīta instrukcija - komanda, piemēram: AND %IX1

0001----------komandas vārda adrese

AND %IX1-----komandas vārds

AND----------operācija (kas ir jāizdara)

%IX1----------operands (ar tā palīdzību operācija tiek izpildīta)

Programmas paraugs	***Programmas tulkojums***
000 LD %IX0	Ja uz ieejas IX0 ir =1
001 AND %IX1	Un uz ieejas IX1 ir =1
002 = %QX0	Tad izejā ir QX0 ir =1
003 LD %IX2	Ja uz ieejas IX2 ir =1
004 OR %IX3	Vai uz ieejas IX3 ir =1
005 = %QX1	Tad izejā QX1 ir =1
…..	
…..	
111 LD %IX4	Ja uz ieejas IX4 ir =1
112 AND %IX5	Un uz ieejas IX5 ir =1
113 AND %IX6	Un uz ieejas IX6 ir =1
114 = %QX2	Tad izejā QX2 ir =1
115 END	Programmas beigas
116	
….	
….	

999 Pēdējā atmiņas līnija

Programmas izskaidrojums:

No 000 līdz 115…..programmas garums.

No 000 līdz 999…viss atmiņas apjoms (programmas garuma maksimums, 1000 rindas).

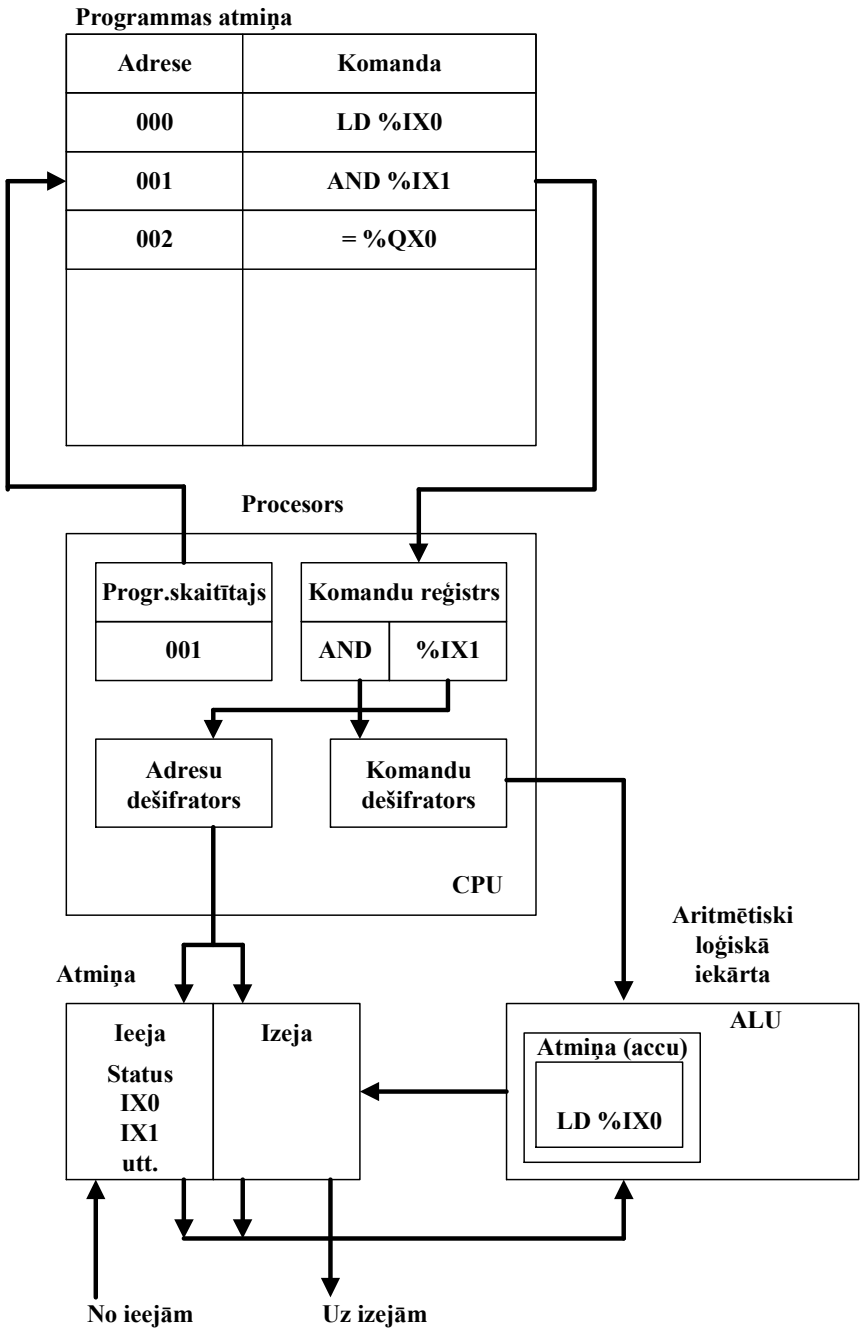
Laiks, kas nepieciešams, lai nolasītu visu atmiņu no 000 līdz 999, tiek saukts par PLC cikla laiku. Šo laiku var atrast PLC instrukcijā.

Ja programmā ir izmantotas pārejas (Jump) instrukcijas, tad var variēt ar programmas izpildes laiku. Tādā gadījumā programmas izpildes cikla laiks ir īsāks nekā PLC instrukcijā dotais.

Jauna cikla sākumā:

Ieeju statuss 1 vai 0 tiks ierakstīti ieejas atmiņā. Turpmākā cikla izpildes laikā ierakstītā informācija par ieejas statusu vairs nemainīsies, pat ja reāli mainīsies pieslēgtās ieejas statuss. Jauna cikla sākumā pieslēgtās ieejas statusa ieraksts atmiņā var mainīties.

Pēc tam ieeju un izeju atmiņas momentānais statuss tiks loģiski apstrādāts. Loģiskās apstrādes uzdevumi ir ierakstīti programmas atmiņā. Apstrāde notiek sērijveidā; pirmie secīgi tiek apstrādāti vārdi ar adresēm no 000 līdz 002. Programmas skaitītājs nodrošina programmas izpildes secīgumu.



2.12. att. **1. shēma**

1. shēma (2.12. att.)

Pieņemsim, ka programmas skaitītājs ir uz vārda adreses 001. Aritmētiski loģiskās apstrādes bloka atmiņā (akumulatorā) (accu) jau ir vārds, kura adrese ir 000. Instrukcija, kas ir ieprogrammēta vārdā ar vārda adresi 001, tagad tiks izpildīta. Vārds AND % IXI tiek nosūtīts uz instrukcijas reģistru. Tiklīdz tas notiek, programmas skaitītājs palielinās tekošo vērtību par 1. Tagad tas uzrādīs adresi 002. Tūlīt pēc esošās instrukcijas AND %IXI izpildes, jauna instrukcija no adreses 002 tiks nosūtīta uz instrukciju reģistru utt.

2. shēma (2.13. att.)

Katrs vārds sastāv no adreses operanda un operācijas.

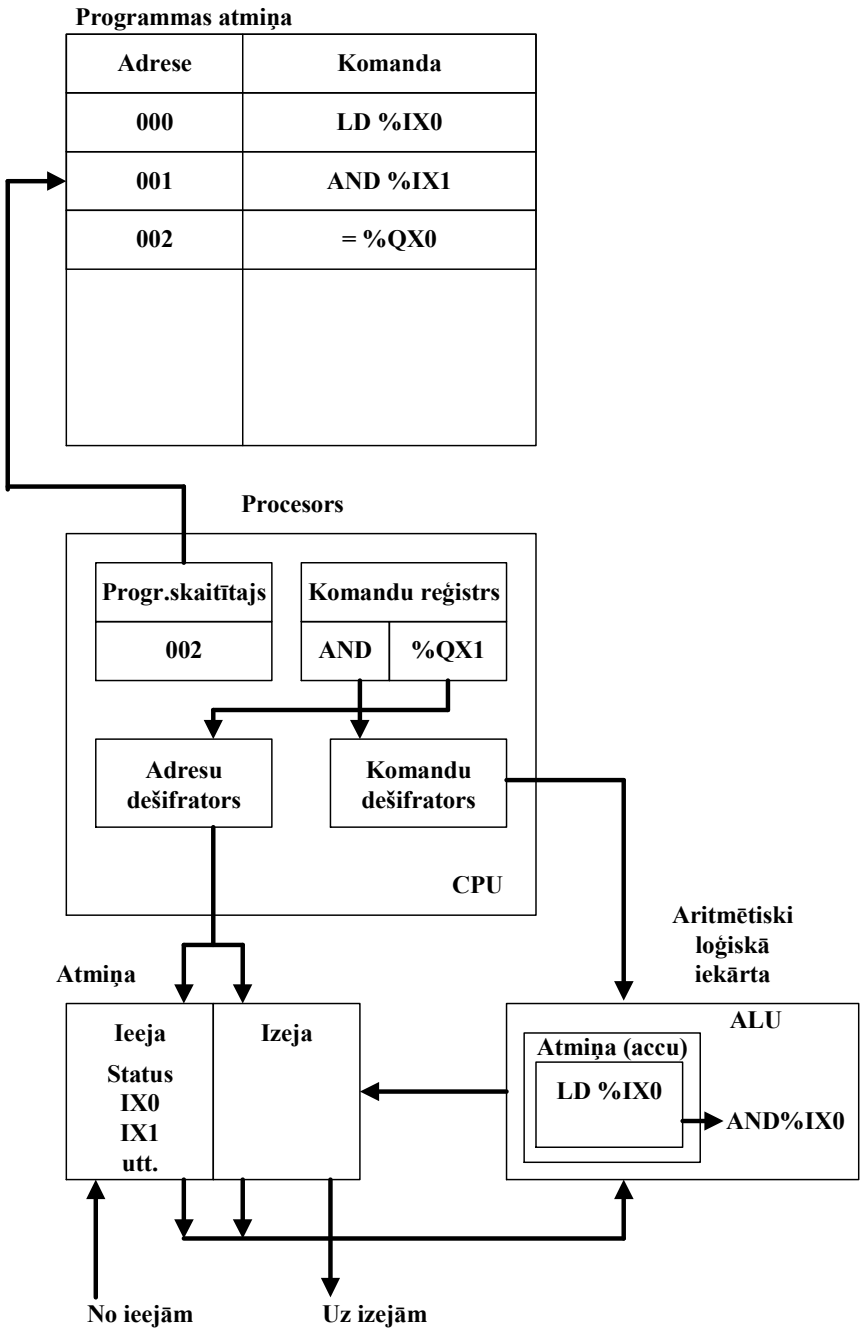
Vārds AND %IXI sastāv no:

- AND.....operācija;
- %IXI.....adreses operands.

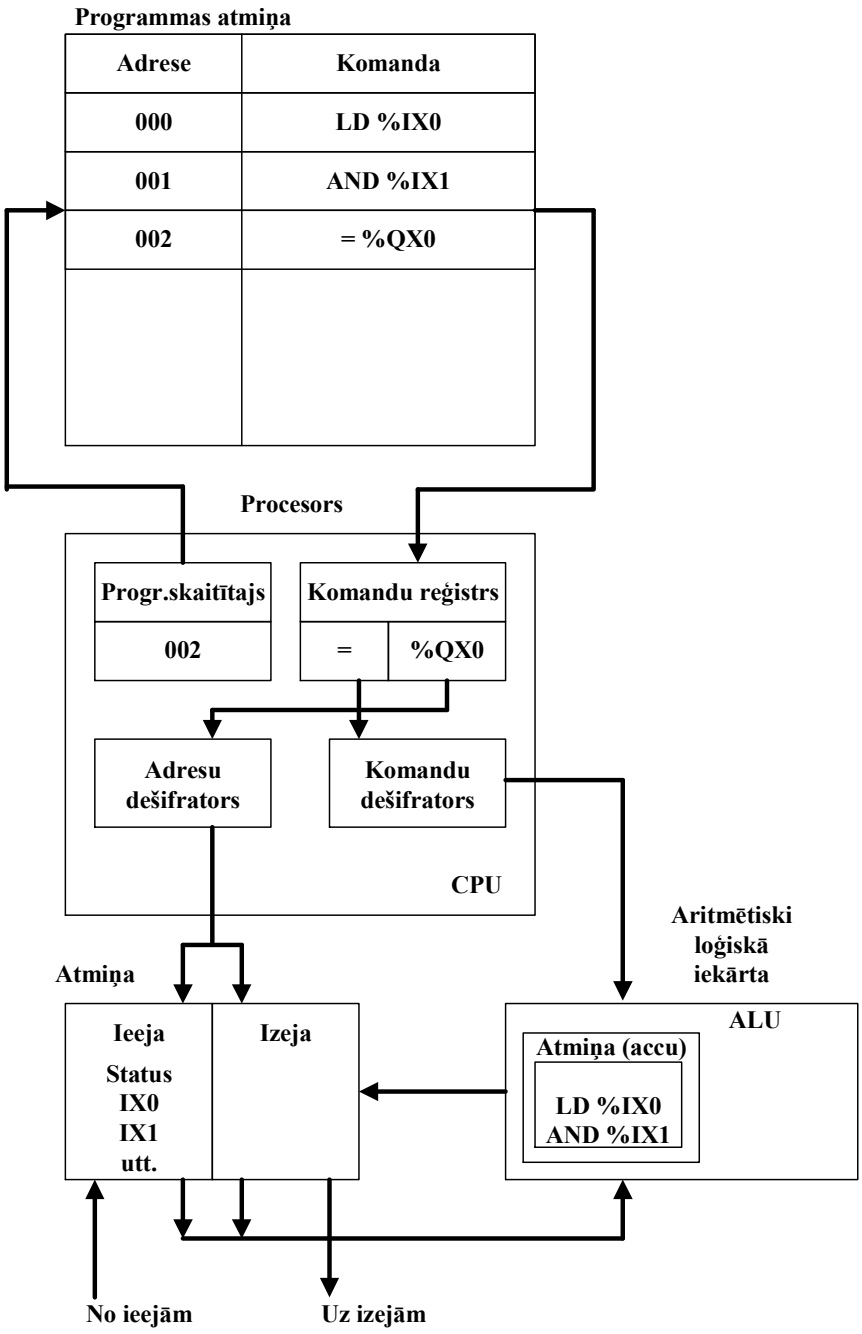
Adreses operands %IXI tiek sūtīts uz adreses dekoderu (dešifratoru), adreses dekoders to izmanto, lai atzīmētu ieejas atmiņas vietu, kur attiecīgās ieejas statuss ir ierakstīts.

%IXI statuss tiek nolasīts no ieejas atmiņas un nosūtīts uz aritmētiski loģiskās apstrādes bloku (ALU).

Operācija tiek nosūtīta uz instrukciju dekoderu. Instrukciju dekoders nosaka, kuras loģiskās funkcijas tiks izpildītas aritmētiski loģiskajā apstrādes blokā (ALU). Operācija AND tiek izpildīta kā loģisks process pie %IXI un aritmētiski loģiskās apstrādes bloka atmiņā (accu) saglabātā iepriekšējās apstrādes rezultāta, šai gadījumā LD %IXO. Šīs operācijas rezultāts LD %IXO AND %IXI ir saglabāts aritmētiski loģiskās apstrādes bloka atmiņā (accu).



2.13. att. **2. shēma**



2.14. att. **3. shēma**

3. shēma (2.14. att.)

Programmas skaitītājs uzrāda vārda adresi 002. Aritmētiski loģiskās apstrādes bloka atmiņā (accu) ir iepriekšējā procesa rezultāts. Instrukcija, kas ieprogrammēta vārdā ar adresi 002, tagad tiek izpildīta. Rezultātā, vārds = % QX0 tiek nosūtīts uz instrukciju reģistru.

Tiklīdz tas ir noticis, programmas skaitītājs palielinās tekošo vērtību par 1, tas rādīs vārda adresi 003. Tūlīt pēc instrukcijas = % QX0 izpildīšanas, instrukcija ar adresi 003 tiks aizsūtīta un ierakstīta instrukciju reģistrā utt.

4. shēma (2.15. att.)

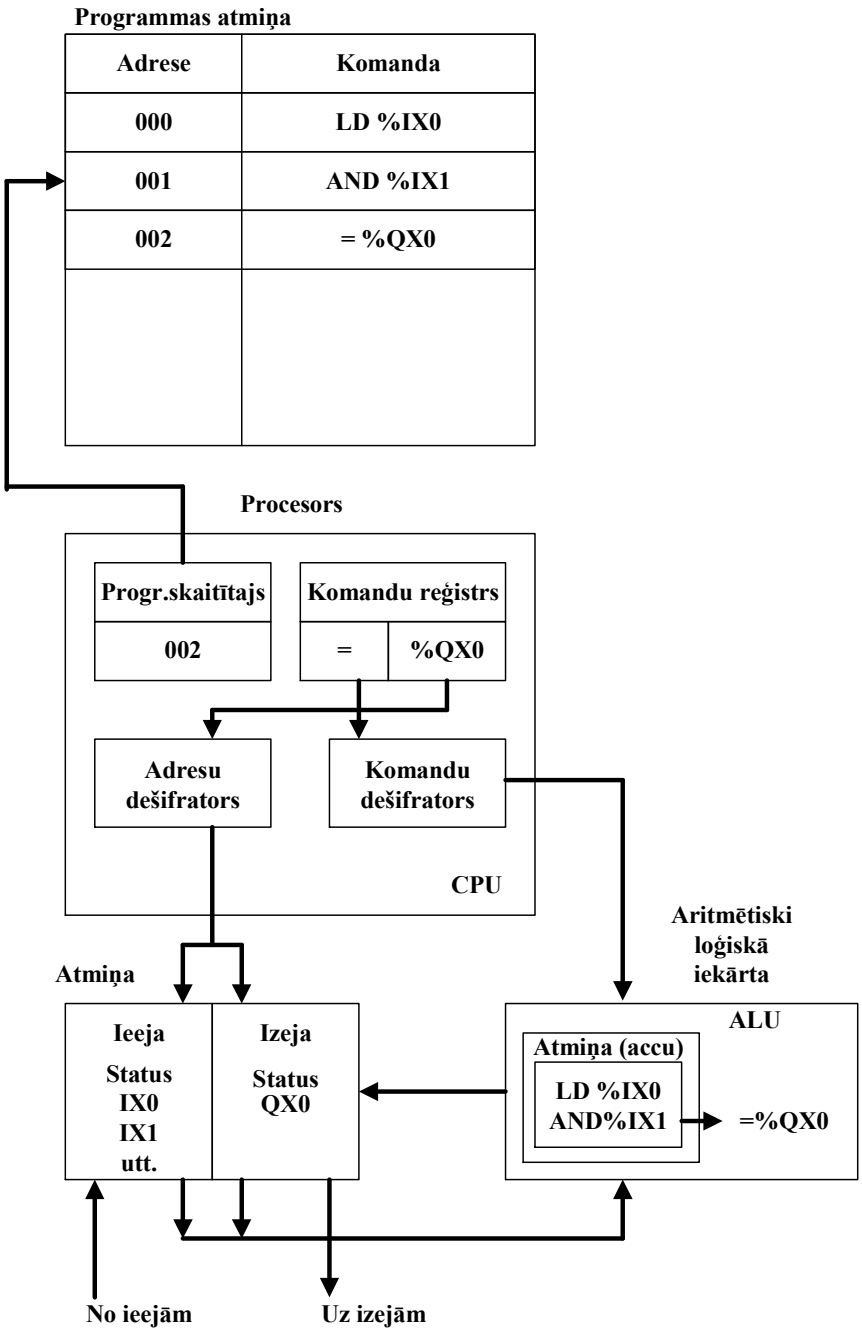
Katrs vārds sastāv no adreses operanda un operācijas.

Vārds = % QX0 sastāv no:

- =.....operācija
- %QX0.....adreses operands

Operācija tiek nosūtīta uz instrukciju dekoderu. Instrukciju dekoders nosaka, kuras loģiskās funkcijas tiks izpildītas aritmētiski loģiskajā apstrādes blokā (ALU). Operācija = tiek izpildīta kā loģisks process un aritmētiski loģiskās apstrādes bloka atmiņā (accu) ir saglabāts iepriekšējās apstrādes rezultāts, šajā gadījumā, LD % IXO AND.%IXI. Šīs operācijas rezultāts ir saglabāts aritmētiski loģiskās apstrādes bloka atmiņā (accu).

Adreses operands %QX0 tiek sūtīts uz adreses dekoderu, un adreses dekoders to izmanto, lai atzīmētu adreses vietu atmiņā, kur glabāsies informācija par attiecīgās izejas statusu. Cikla beigās, šī procesa darbības rezultāts, kas saglabāts aritmētiski loģiskās apstrādes bloka atmiņā, tiek pārnests uz konkrētajai izejai atbilstošo atmiņas adresi. Izeja atkarībā no attiecīgās atmiņas vietas statusa vēlāk tiks ieslēgta vai izslēgta.



2.15. att. **4. shēma**

3. PROGRAMMĒŠANAS STANDARTS IEC 61131-3

3.1. Standarta struktūra

Starptautiskā elektrotehniskā komisija (IEC) 1993. gadā pieņēma standartu IEC 61131 par PLC programmēšanas valodām. Tas tika darīts ar nolūku piespiest ražotājus PLC programmēšanai izmantot kopējas valodas, lai atvieglotu speciālistu apmācību, radītu iekārtu savstarpējo aizvietojamību, kā arī ļautu izmantot kopēju programmnodrošinājumu vairāku ražotāju produkcijai.

IEC 1131-3 ir pirmais īstais līdzeklis programmu valodu standartizēšanai industrialās automātikas vajadzībām. Pateicoties starptautiskajam atbalstam, šis standarts nav atkarīgs ne no vienas kompānijas.

IEC 61131 un DIN EN 61131 ir jaunākā versija agrāk izstrādātajam IEC 1131 standartam. Tas sastāv no:

- 1.Daļa: vispārējais apskats;
- 2.Daļa: aparatūra, prasības un pārbaudes;
- 3.Daļa: programmēšanas valodas;
- 4.Daļa: lietotāja rokasgrāmata;
- 5.Daļa: komunikācijas, komunikāciju specifikācijas.

Aplūkojot šī standarta trešo daļu jāsaka, ka tur ieguldīts liels darbs:

- Task Force 3 rezultāts, programmējošās valodas IEC TC65 SC65B ietvaros;
- 7 starptautisku kompāniju desmitiem gadu ilga darba un pieredzes rezultāts industriālās automātikas sfērā;
- apmēram 200 lappuses teksta ar 60 tabulām, ieskaitot parametru īpašību tabulas;
- programmējošo valodu sintakses un semantikas īpatnības, ieskaitot vispārēju valodas struktūru un programmu modeli.

Ir iespēja apskatīt standartu IEC 61131-3 sadalot divās daļās.

1. Vispārīgie programmēšanas valodu elementi
2. Programmēšanas valodas

Aplūkosim šīs daļas detalizētāk.

3.2. Vispārīgie programmēšamas valodu elementi

Datu tipi

Datu tipi ir noteikti un tie atrodami standartā pie vispārīgajiem elementiem. Izvairīties no kļūdām jau agrīnā programmēšanas stadijā ļauj datu tipu šķirošana. To izmanto, lai noteiktu ikviena programmā lietotā parametra datu tipu. Tas pasargā, piemēram, no datuma dalīšanas ar skaitli un tamlīdzīgām kļūdām.

Vienkāršakie datu tipi ir: loģiskie BOOL (Boolean) 0;1, veseli skaitļi INT (Integer) no -32768 līdz +32767, reāli skaitļi REAL – skaitļi ar peldošu komatu, baiti BYTE – 8 bitu rinda un vārdi WORD – 16 bitu rinda, kā arī datums (Date), laika intervāls TIME un rinda STRING – mainīga garuma bitu rinda. Pamatojoties uz minēto, lietotājam ir iespējams noteikt personiskos datu tipus, kas pazīstami kā otrreizējie datu tipi. Tādējādi iespējams noteikt analogās ieejas kanāla datus kā informācijas tipu, ko pēc tam var daudzkārt atkārtoti izmantot.

Mainīgie

Mainīgie ir paredzēti, lai apzīmētu aparatūras adreses (ieejas un izejas) konfigurācijās, resursos vai programmās. Tādējādi tiek panākta augsta līmeņa neatkarība no aparatūras nodrošinājuma (hardware) un iespējams radīt daudzkārt izmantojamu programmatūru.

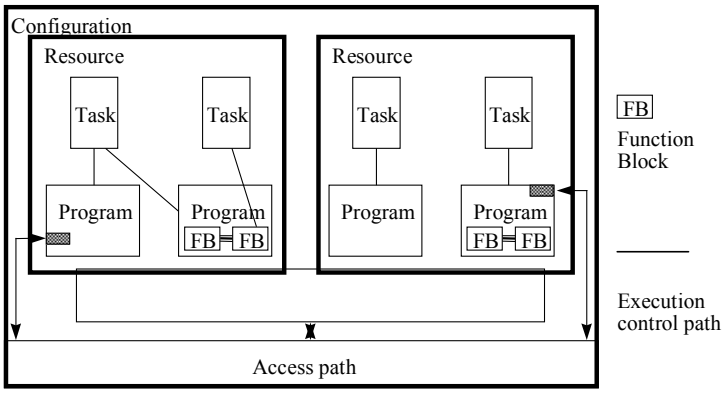
Mainīgo pielietojumu parasti ierobežo bloks, kurā tie tiek izmantoti, tiem ir vietējs pielietojums. Tas nozīmē, ka mainīgie var tikt atkārtoti izmantoti citos blokos bez bailēm, ka radīsies kļūdas. Ja mainīgos nepieciešams izmantot visā projektā, par tiem attiecīgi jāpaziņo, ka tie ir globālie, piemēram, (VAR_ GLOBAL). Lai panāktu pareizus iestatījumus, parametriem būtu jāuzdod sākotnējā vērtība gan uzsākot, gan atsākot darbības.

Konfigurācija, resursi un uzdevumi

Lai labāk izprastu šos terminus, aplūkosim programmnodrošinājuma modeli tā, kā definēts standartā (skat. 3.1. att.).

Augstākajā līmenī viss programmnodrošinājums, kas paredzēts noteiktas kontroles problēmas risināšanai, var tikt formulēts kā *konfigurācija.* Katras vadības sistēmas konfigurācija ir īpatnēja, ieskaitot iekārtas konstruktīvo izvietojumu, datu apstrādes resursus, atmiņu adreses I/O kanāliem un sistēmas iespējas.

Konfigurācijas ietvaros iespējams definēt vienu vai vairākus *resursus*. Var pieņemt, ka *resursi* ir datu apstrādes iespēja, ko spēj izpildīt IEC programmas.



3.1. att. **Programmas modeļa struktūra pēc IEC 1131-3**

Resursā var būt definēti viens vai vairāki *uzdevumi*. *Uzdevumi* vada programmu vai funkciju bloku izpildi. Tie var tikt izpildīti periodiski, vai gadījumā, kad ir izmainīta uzdotā mainīgā lieluma vērtība.

Programmas tiem izveidotas no vairākiem programmnodrošinājuma elementiem, kas rakstīti jebkurā no IEC valodām. Parasti programma sastāv no funkciju tīkla un *funkciju blokiem*, kuros var izmainīt doto informāciju. Funkcijas un funkciju bloki ir programmas veidošanas pamatelementi, kas satur informācijas struktūru un algoritmu.

Salīdzināsim šo ar tradicionālo PLC: tas satur vienu resursu, izpilda vienu uzdevumu, vada pēc vienas programmas un to izpilda noslēgtā ciklā. IEC 61131-3 vēl

papildina šo modeli nosakot turpmāko attīstību nākotnē. Piemēram, daudzprocesoru datu apstrāde un tā tālāk. Šī nākotne nav aiz kalniem. Pietiek aplūkot sadalītās skaitļošanas sistēmas vai kontroles sistēmas, kas darbojas reālā laikā. IEC 61131-3 ir piemērots lai risinātu plaša spektra praktiski pielietojamus uzdevumus bez nepieciešamības apgūt papildus programmu valodas.

Programmu moduļi

Programmas moduļi ir programmas, funkciju bloki un funkcijas, kas standartā IEC 61131-3, tiek sauktas par programmu moduļiem.

Funkcijas

IEC ir noteiktas standarta funkcijas un lietotāja noteiktās funkcijas. Standarta funkcijas ir, piemēram, ADD - saskaitīšana, ABS - absolūtais lielums, modulis, SQRT - kvadrātsakne, SIN - sinuss un COS - kosinuss. Lietotāja funkcijas var tikt izmantotas daudzkārtēji, bet tās iepriekš ir jādefinē.

Funkciju bloki, FB

Funkciju bloki ir ekvivalenti mikroshēmām, kas izpilda noteiktu vadības funkciju. Tie satur gan informāciju, gan algoritmu un tadējādi var atkārtot iepriekšējo darbību. Tiem ir konkrēti noteikts interfeiss un ir slēptas iekšējās īpašības, līdzīgi, kā, piemēram, mikroshēmām, vai “melnajai kastei”. Šai sakarā ir liela atšķirība starp dažāda līmeņa programmētājiem un remonta un apkopes personālu.

Temperatūras regulēšanas kontūrs ar PID - proporcionāli integrāli diferenciālo regulatoru ir lielisks funkciju bloka piemērs. Vienreiz definēts, tas var tikt lietots atkal un atkal, gan tajā pašā programmā, gan dažādās programmās, pat dažādos projektos. Tie ir augsta līmeņa daudzkārt izmantojamie funkciju bloki.

Funkciju bloki var būt uzrakstīti jebkurā IEC valodā, lielākoties pat "C". Tādējādi tos var definēt lietotājs. No jauna veidotie funkciju bloki pamatojas uz FB standarta blokiem, bet tai pašā laikā tie ir pilnīgi atšķirīgi. Standarts pieļauj izveidot un piedzīt funkciju bloku noteikta uzdevuma izpildei, standarts nosaka tikai struktūru.

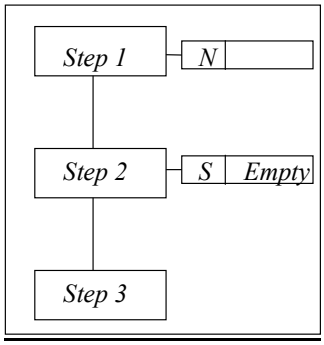
Programmas

No iepriekš minētās informācijas par pamatsastāvdaļām varam secināt, ka programma savstarpēji saistīta struktūra, kas veidota no funkcijām un funkciju blokiem. Programmu var uzrakstīt jebkurā no noteiktajām programmēšanas valodām.

Secīgo funkciju diagramma

Secīgo funkciju diagramma SFC grafiski apraksta kontroles programmas secīgo darbību. Valoda SFC strukturizē programmas iekšējo organizāciju un palīdz sadalīt kontroles problēmu viegli vadāmās daļās.

SFC sastāv no soļiem (Steps), kas savienoti darbības blokos (Action Blocks) un pārejām (Transitions). Katrs solis norāda uz kontrolējamās sistēmas īpašo stāvokli. Pāreja ir saistīta ar nosacījumu, ja tekošais solis, vai nosacījums ir izpildīts (true), tad ļauj pāriet uz nākošā programmas soļa izpildi. Soļi ir saistīti ar noteiktu darbību blokiem, kas izpilda konkrētu vadības programmas funkciju. Katrs programmas elements var tikt ieprogrammēts jebkurā no IEC valodām, ieskaitot SFC.



3.2. att. **SFC programmas fragments**

Var izmantot gan alternatīvo secīgumu, gan paralēlo secīgumu, tos parasti izmanto pielietojamajās programmās. Piemēram, viena secība tiek izmantota galvenajam procesam, bet otra, lai vadītu vispārējos operatīvos ierobežojumus.
Vienkāršās struktūras dēļ SFC iespējams izmantot arī kā komunikācijas instrumentu, apvienojot dažādu aprindu cilvēkus, valstis un reģionus.

3.3. Programmēšanas valodas

PLC programmēšanas valodas

Pēc standarta IEC 61131 pēdējās 3 redakcijas tiek izmantotas 5 PLC programmēšanas valodas. Valodu sintakse un semantika ir iepriekš noteiktas, neatstājot iespēju nekādu dialektu izmantošanai. Iemācoties šīs valodas, Jūs variet tās izmantot visās plaši izplatītajās sistēmās, kas pamatojas uz šo standartu.

Ir trīs grafiskās un divas tekstuālās valodas.

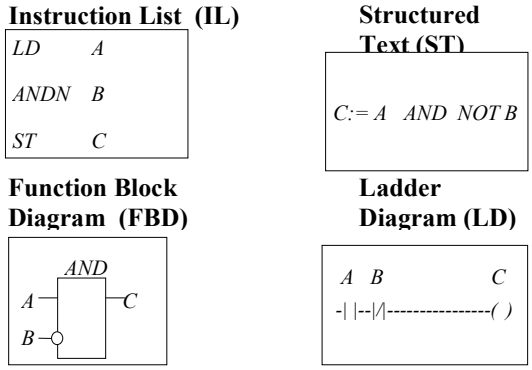
Tekstuālās valodas:

- Instrukciju saraksts, IL (Instruction List);
- Strukturētais teksts, ST (Structured Text);

Grafiskās valodas:

- Kāpņu diagramma LD (Ladder Diagram) (Releju diagramma);
- Funkciju bloku diagramma, FBD (Function Block Diagram);
- Secīgā funkciju diagramma SFC (Sequential Function Chart).

3.3. attēlā visas 4 valodas apraksta vienu un to pašu vienkāršo programmas fragmentu.



3.3. att. **Programmēšanas valodas**

Programmēšanas valodas izvēle ir atkarīga no:

- programmētāja priekšzināšanām un sagatavotības;

- risināmās problēmas sarežģītības;
- problēmas apraksta līmeņa;
- vadības sistēmas struktūras;
- tīkla interfeisa.

Visas valodas ir savstarpēji saistītas, tām ir kopīgs instrumentu komplekts.

- *Kāpņu diagramma (Releju diagramma)* (LD) nāk no ASV. Tā pamatojas uz Relay Ladder Logic grafisko shēmu izmantošanu.
- *Instrukciju saraksts* (IL) ir tās Eiropas analogs. Tekstuāla valoda.
- *Funkciju bloku diagramma* (FBD) ir ļoti pierasta lieta apstrādes industrijā. Tā atklāj funkciju, funkciju bloku un programmu darbību kā savstarpēji vienotus grafiskos blokus, tāpat, kā, piemēram, elektronisko ķēžu diagrammās. Tā izprot sistēmu kā signālu plūsmu starp apstrādes elementiem.
- *Strukturētais teksts* (ST) ir ļoti spēcīga valoda, kas nāk no Ada, Pascal un C. To var lieliski izmantot, lai definētu kompleksus funkciju blokus, kas var tikt lietoti ikvienā citā valodā.

Programmēšana no augšas uz leju un no apakšas uz augšu

Standarts ļauj veidot programmu divējādi: no augšas uz leju, vai no apakšas uz augšu.

Jūs apskatāt visu programmu kopumā un sadaliet to apakšprogrammās, uzdodiet mainīgos lielumus utt. Vai arī sākat programmēšanu no pašiem pamatiem, piemēram, izveidojot funkcijas un funkciju blokus. Lai ko Jūs arī izvēlētos, programmas izstrādes vide palīdzēs visā procesa laikā.

Izpildījums

Pilnīgi visas standarta IEC 61131-3 prasības nav viegli izpildīt, tāpēc standarts pieļauj ieviest arī atsevišķas atšķirīgas izstrādnes. Tas attiecas uz vairākām papildus valodām, funkcijām un funkciju blokiem. Tas dod piegādātājam zināmu rīcības brīvību, bet lietotājam par to ir jāzina analizējot tehnoloģisko procesu.

Daudzas esošās IEC programmēšanas vides piedāvā gandrīz visu nepieciešamo, ko vien var gribēt no modernajām programmēšanas vidēm: darbs ar peli, grafiskos programmēšanas attēlus, daudzkārtējus "logus", iebūvētas hiperteksta funkcijas, dažādu operacionālo sistēmu uzturēšana, saglabāšanu darba laikā u.c. Tomēr atcerieties, tas nav paredzēts pašā standartā, tā ir viena no piegādātāja variāciju iespējām.

Noslēgums

IEC 61131-3 standarta tehniskā nozīme ir ļoti liela. Turklāt pastāv arī attīstības un specializācijas iespējas.

IEC 61131-3 ir būtiska ietekme uz rūpnieciskajām vadības sistēmām dažādās industrijas nozarēs. Šī standarta darbība nenorobežojas tikai ar PLC tirgu, bet gan izplatās softlogic sistēmu un uz personālajiem datoriem balstītu kontroles sistēmu tirgu, ieskaitot SCADA paketes, utt.

Plaša šī standarta izmantošana dažādos pielietojuma virzienos dod būtisku labumu lietotājam un programmētājam.

Standarta izmantošanas priekšrocības ir atkarīgas no pielietojuma sfēras, nosauksim dažas:

- samazinās laika patēriņš personāla apmācībai par programmu palaišanu, mašīnu iestatīšanu, apkopšanu un konsultācijām;
- vienkāršojas programmēšana, rezultātā, samazinās iespēja kļūdīties;
- plaši izmantojamu programmēšanas tehnisko paņēmienu pielietošana;
- iespēja apvienot dažādus komponentus no dažādiem ražotājiem, projektiem, vietām, uzņēmumiem.

4. PLC IEBŪVĒŠANA UN PIESLĒGŠANA

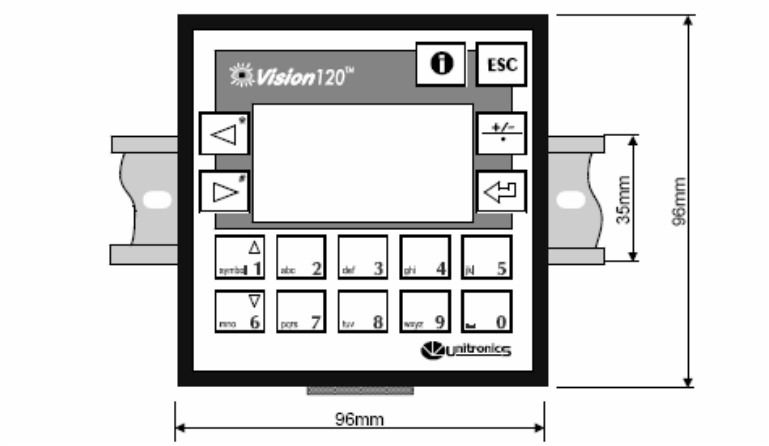
Montējot un pievienojot PLC, jāatrisina sekojošie jautājumi:

1. PLC montēšana uz paneļa;
2. barošanas sprieguma pievienošana PLC;
3. sensoru pievienošana PLC ieejām;
4. izpildierīču pievienošana PLC izejām.

4.1. PLC montēšana uz paneļa

Pirms izvēlēties atbilstošu paneli, ir jāievērtē sekojošie faktori:

- mehānisko bojājumu iespējamība un to apjoms;
- mitrums;
- putekļi, korozija u.c.
- ķimikāliju iedarbība;
- dzesēšana.



4.1. att. **Kontrollera montāža uz DIN sliedes**

Panelim efektīvi jāaizsargā PLC pret mehāniskiem bojājumiem. Vispiemērotākais šim nolūkam ir metāla korpuss. Tomēr metāla korpusam piemīt trūkums - darba laikā nav iespējams vērot ieeju un izeju statusu. Risinājumam tiek piedāvāts izturīgas, caurspīdīgas plastmasas korpuss.

PLC novietojumu šajā korpusā nosaka nepieciešamība nodrošināt normālu dzesēšanu un nepieļaut citu, šai korpusā izvietoto, ierīču traucējošo signālu iedarbi.

Dzesēšana

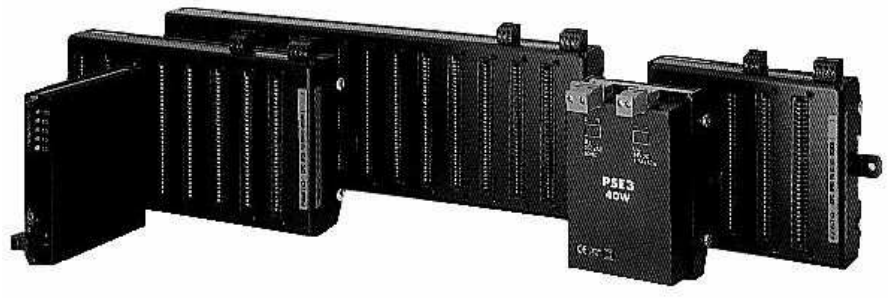
PLC darbības laikā izdalās siltums, tāpēc ir izveidotas ventilācijas atveres, kuras montējot vadības bloku nedrīkst aizvērt. Tam ir jāpievērš liela uzmanība, un PLC ir jāinstalē korpusā tādā veidā, lai saglabātu dabisko ventilāciju.

Traucējumu signāli

Izvēloties metāla korpusu, ir iespējams likvidēt ārējo traucējumu signālu ietekmi. Tomēr komponenti, kas atrodas korpusa iekšienē joprojām var radīt savstarpēji iedarbīgus traucējumu signālus. Tāpēc uzmanība jāpievērš augstas frekvences kontrolleru montāžai. Noteikti jāizlasa komponentu montāžas instrukciju. Ļoti svarīga ir vadu un kabeļu montāža. Traucējumu signāli var iedarboties uz sensoru un operatīvo regulatoru savienojumiem ar PLC ieejām, tādā pakāpē, ka ieejā parādās 1.

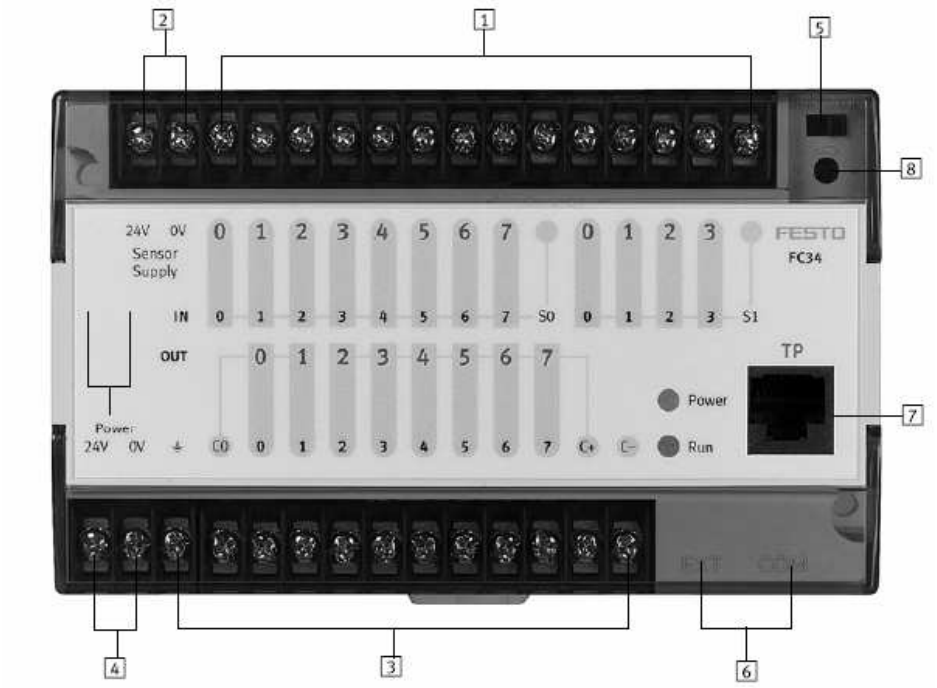
4.2. Barošanas sprieguma pievienošana PLC

Moduļu tipa PLC, barošanas spriegums tiek pievienots barošanas modulim. Citi moduļi tiek apgādāti ar strāvu kopnes („bus") sistēmas ietvaros. Šim nolūkam lieto speciālu kopnes paneli, kurā izvietotas un savstarpēji savienotas daudzas spraudņu ligzdas.



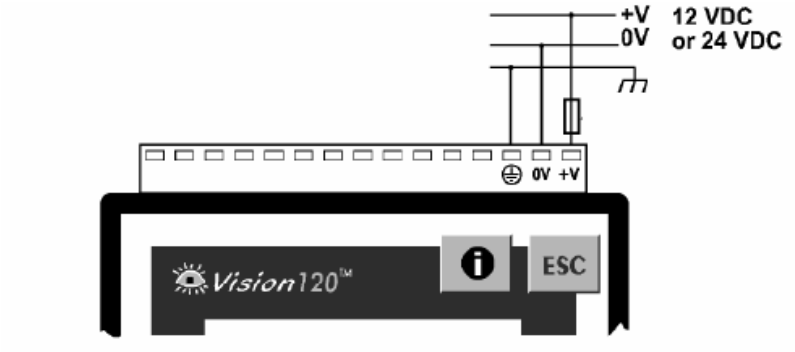
4.2. att. **Kopnes paneļa izpildījums**

Barošanas spriegumu kompaktam PLC jāpievieno pie savienojošajām spailēm, kas paredzētas šim nolūkam (skat. 4.3.att.).



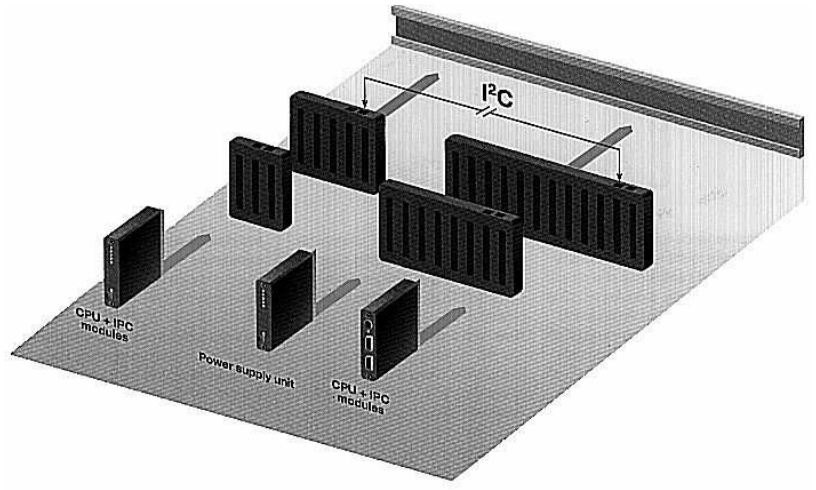
4.3. att. **Kompaktais loģiskais kontrollers IPC FEC FC34:**
1 – divpadsmit diskrētās ieejas; 2 – atsevišķas sensoru barošanas spailes; 3 – astoņas diskrētās izejas;
4 – barošanas spailes; 5 – slēdzis RUN/STOP; 6 – divi virknes interfeisi; 7 – LAN datortīkls;
8 – digitāls potenciometrs ar izšķirtspēju 0...63.

Barošanas bloka uzdevums ir pārveidot pienākošo barošanas spriegumu dažādos spriegumos, kas nepieciešami PLC un sensoru darbināšanai.



4.4 . att. **Barošanas ķēdes slēgums Vision 120 kontrollerim**

Pirms PLC pievienošanas spriegumam, jāiepazīstas ar PLC tehnisko aprakstu. Barošanas spriegums ne vienmēr ir 24V līdzstrāva, tas var būt arī 230V maiņstrāva.



4.5. att. **Barošanas moduļa savienojums PLC modulārā sistēmā**

Drošības apsvērumu dēļ PLC jābūt iezemētam. Iezemēšana kalpo ne tikai drošībai, bet arī kā traucējumu ierobežotājs.

4.3. Sensoru pievienošana PLC ieejām

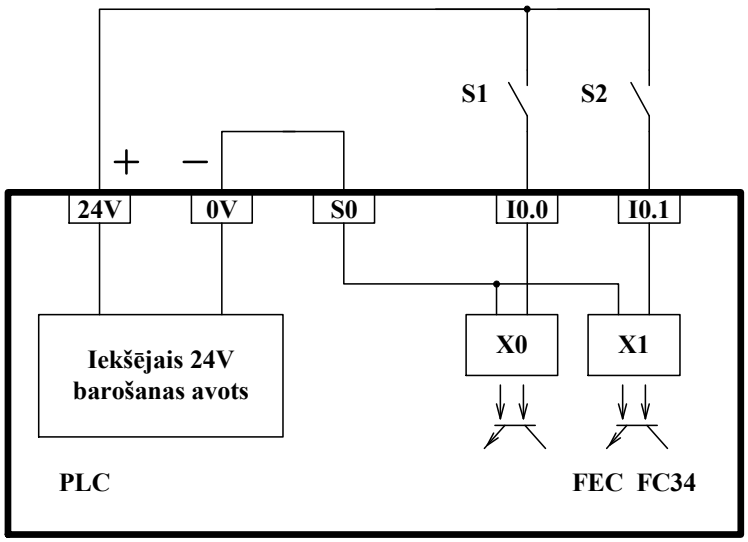
PLC digitālās ieejas reaģē uz signāla esamību vai neesamību. Atkarībā no PLC ieeju slēguma konfigurācijas, loģiskā vienlnieka signāls var būt augsts sprieguma līmenis (uz ieeju padod +24V), vai arī pretēji – zems sprieguma līmenis (ieeju saslēdz ar – masu 0V, vai -24V).

Ieejas statuss ir atkarīgs no signāla esamības:

- Ja uz PLC ieejas ir signāls, ieejas statuss 1;
- Ja uz PLC ieejas nav signāla, ieejas statuss 0.

Par ieejas statusu norāda gaismas diode (LED):

- Gaismas diode deg ---- statuss 1;
- Gaismas diode nedeg ---- statuss 0.



4.6. att. **Sensoru slēgums ar kopīgo mīnusu pie iekšējā barošanas avota**

Par ieejas signālu visbiežāk izmanto 24 V līdzstrāvu. Ja uz ieeju padod:

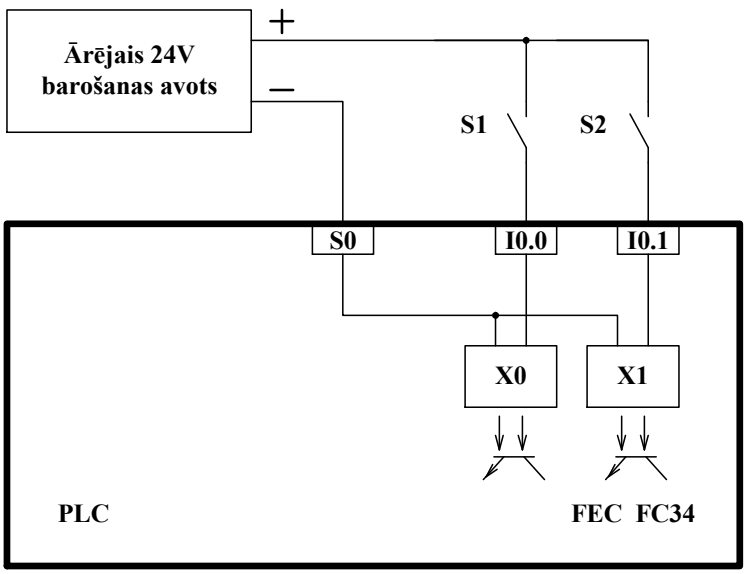
- 24V ---- gaismas diode deg ---- ieejas statuss 1;

- 0V ---- gaismas diode nedeg ---- ieejas statuss 0.

24 V līdzstrāvu uz ieejām padod no PLC iekšējā barošanas avota, vai izmantoto ārējo barošanas avotu.

Pašreizējās paaudzes PLC ieejas apzīmē saskaņā ar IEC 61131 standartu ar X0, X1, X2 utt. Katrai PLC ieejai ir divi pieslēguma punkti (skat. 4.6. att.).

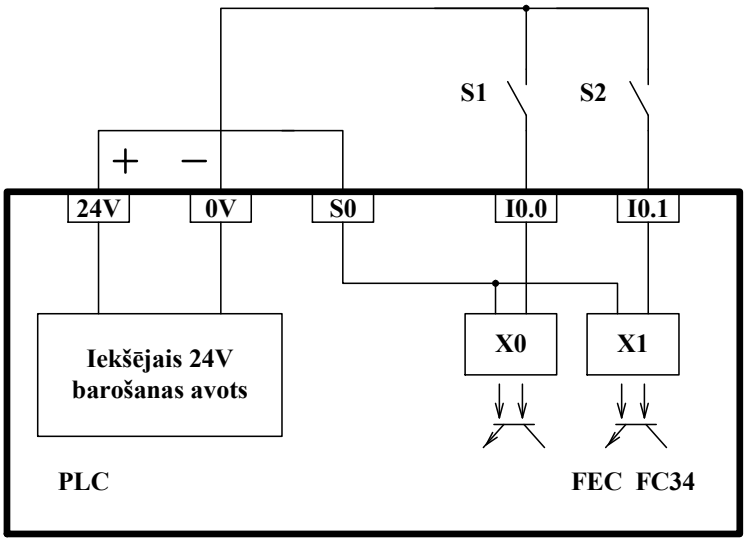
Kontrolleriem, kura ieejas reaģē uz +24 V līdzstrāvu, visu ieeju viens pieslēguma punkts ir pastāvīgi pievienots līdzstrāvas avota mīnusam. Šādu slēgumu sauc par kopīgā mīnusa slēgumu. Shēma ieeju pieslēgšanai PLC ar iekšējo barošanas avotu, ar kopīgo mīnusu, parādīta 4.6. attēlā. Katras ieejas otra spaile caur sensoru tiek pievienota 24 V avota plusa spailei.
Sensors ieslēdz plusu.



4.7. att. **Sensoru slēgums ar kopīgo mīnusu pie ārējā barošanas avota**

Shēma ieeju pieslēgšanai PLC ar ārējo barošanas avotu ar kopīgo mīnusu parādīta 4.7. attēlā. Otra ieejas spaile caur sensoru ir pievienota +24 V līdzstrāvai. Sensors ieslēdz plusu.

PLC, kura ieejas reaģē uz saslēgšanu ar barošanas avota mīnusa spaili (parasti ar masu, 0 V), visu ieeju viens savienojošais punkts ir pastāvīgi pievienots 24 V līdzstrāvas avota plusa spailei. Šādu slēgumu sauc par kopīgā plusa slēgumu.



4.8. att. **Sensoru slēgums ar kopīgo plusu pie iekšējā barošanas avota**

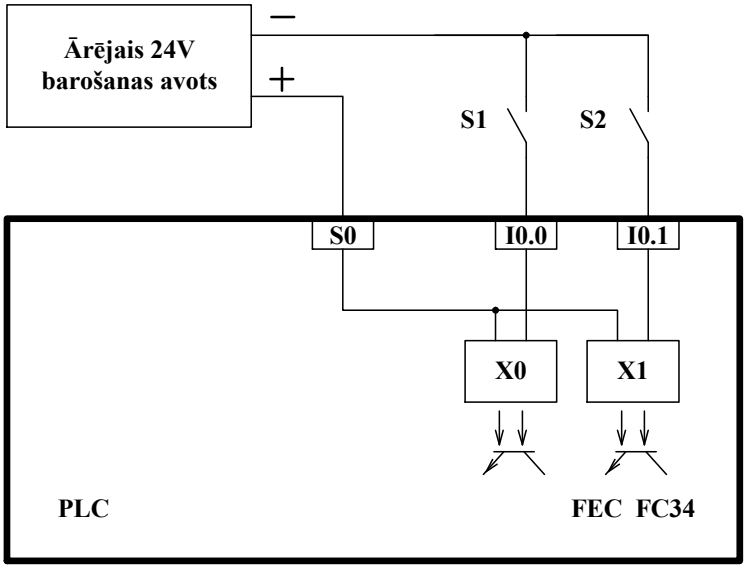
Shēma sensoru pieslēgšanai PLC ar iekšējo barošanas avotu un ar kopīgo plusu parādīta 4.8. attēlā. Otra ieejas spaile caur sensoru ir pievienota 24 V avota mīnusa spailei.

Sensors ieslēdz mīnusu.

Shēma sensoru pieslēgšanai PLC ar ārējo barošanas avotu un ar kopīgo plusu parādīta 4.9. attēlā. Otrā ieejas spaile caur sensoru ir pievienota 24 V avota mīnusa spailei.

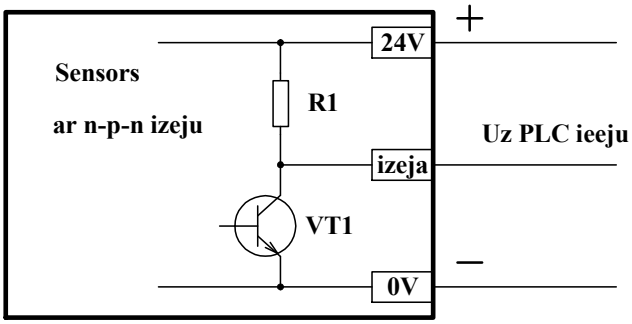
Sensors ieslēdz mīnusu.

Sensoriem, kam izejā ir parasti savienojošie vai atvienojošie kontakti, nav problemātiski pievienot plusu vai mīnusu, barošanas ķēdes polaritātei nav nozīmes.



4.9. att. **Sensoru slēgums ar kopīgo plusu pie ārējā barošanas avota**

Tiem sensoriem, kuru izejas slēdža daļa sastāv no tranzistora, ir ļoti svarīgi, vai pievienots ir mīnuss vai pluss. Par iemeslu tam ir divi tranzistoru tipi: NPN vai PNP, kurus pieslēdzot ir svarīgi ievērot polaritāti.

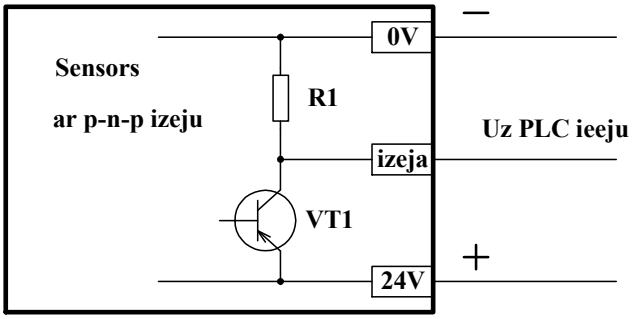


4.10. att. **Sensora ar NPN izeju slēguma shēma**

Vienkārša principiālā elektriskā shēma ar NPN-tranzistora kopemitera slēgumu dota attēlā 4.10. Kamēr tranzistors ir aizvērts uz sensora izeju, caur rezistoru R1, tiek

padots spriegums no +24 V barošanas ķēdes. Tiklīdz sensors nostrādā, tranzistors VT1 atveras un saslēdz sensora izeju ar 0 V.

NPN sensora izejai tranzistors ieslēdz mīnusu.

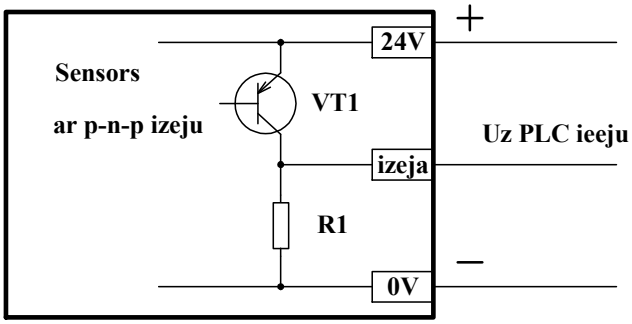


4.11. att. **Sensora ar PNP izeju slēguma shēma**

Vienkārša principiālā elektriskā shēma ar PNP-tranzistora kopemitera slēgumu redzama attēlā 4.11. Kad sensors nostrādā, tranzistors VT1 atveras un padod spriegumu no +24 V barošanas ķēdes uz sensora izeju.

PNP sensora izejai tranzistors ieslēdz plusu.

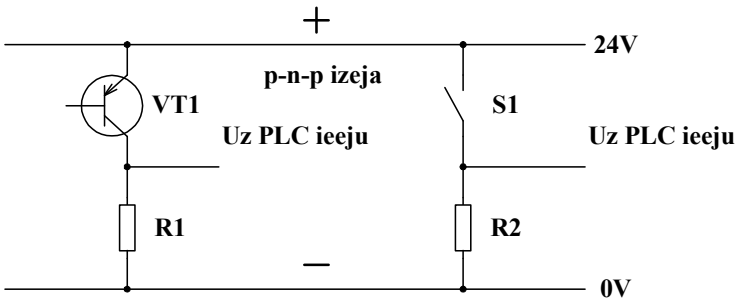
4.11. attēlā doto slēguma diagrammu parasti zīmē spoguļattēlā (skat. 4.12. att.).



4.12. att. **Sensora ar PNP izeju slēguma shēma**

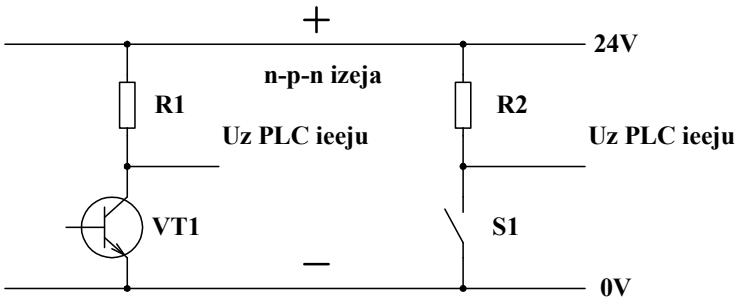
Ja par slēdzi izmantojam tranzistoru, tad iegūstam bezkontaktu shēmu (skat. 4.13. att.). Pilnīgi atvērtu (vadošu) tranzistoru var salīdzināt ar slēdzi, kura kontakti ir saslēgti. Tas savieno barošanas plusu ar PLC ieeju.

PNP-tranzistora slēdža un kontaktu slēdža principiālās shēmas parādītas attēlā 4.13.



4.13. att. **PNP tranzistora un slēdža salīdzinājums**

Gan tranzistors VT1, gan slēdzis S1 ieslēdz barošanas avota plusu. PLC ieejām ir jābūt kopējā mīnusa slēgumā.



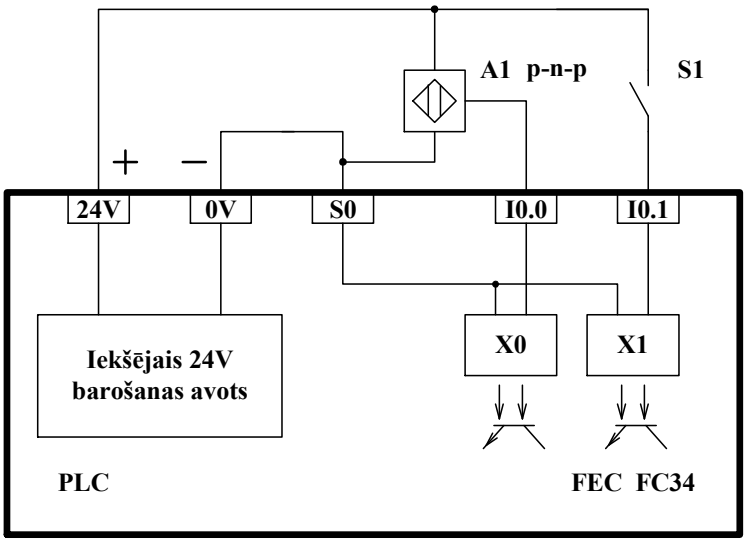
4.14. att. **NPN tranzistora un slēdža salīdzinājums**

NPN- tranzistora slēdža un kontaktu slēdža principiālās shēmas parādītas attēlā 4.14. Gan tranzistors VT1, gan slēdzis S1 ieslēdz barošanas avota mīnusu. PLC ieejām ir jābūt kopējā plusa slēgumā.

PNP tipa sensoru pieslēgšana PLC ieejām un iekšējam barošanas avotam dota principiālajā shēmā 4.15. attēlā. PLC ieejām ir jābūt kopējā mīnusa slēgumā.

Kontrollera ieeju kopīgo spaili S0 savieno ar barošanas avota mīnusa spaili. Pieslēdzot sensora A1 barošanas spailes, jāievēro polaritāte.

Ja sensors A1 nostrādā, tas padod spriegumu no +24V līnijas uz PLC ieejas otro spaili I0.0. *PNP sensors ieslēdz plusu*.

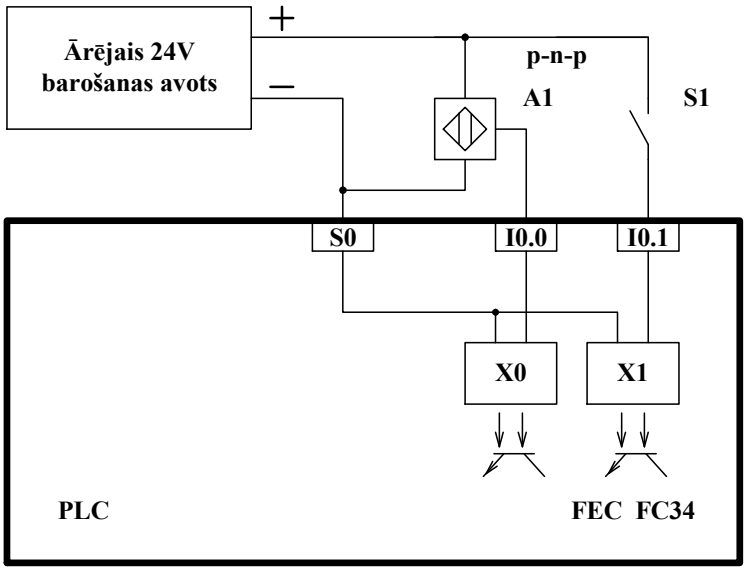


4.15. att. **PNP sensoru pievienošana PLC ar iekšēju barošanas avotu**

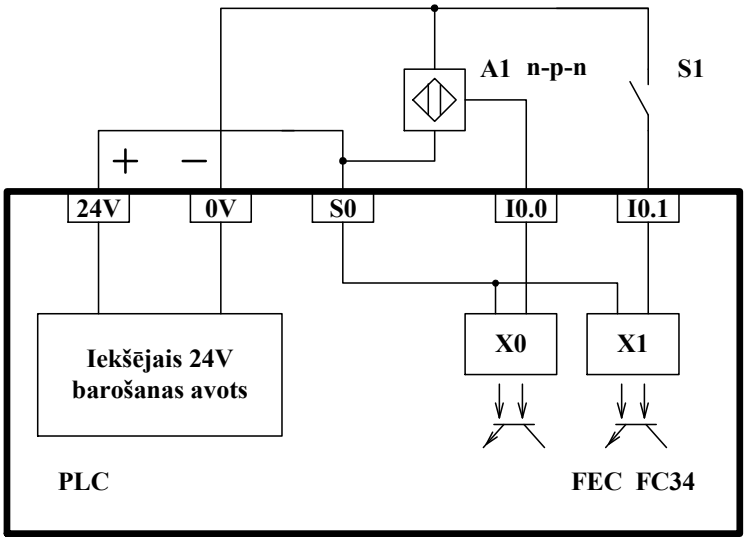
PNP tipa sensoru pieslēgšana PLC ieejām un ārējam barošanas avotam redzama principiālajā shēmā 4.16. attēlā. PLC ieejām ir jābūt kopējā mīnusa slēgumā. Ja PNP sensors nostrādā, tas padod spriegumu no +24 V līnijas uz PLC ieejas otro spaili I0.0. *PNP sensors ieslēdz plusu*.

NPN tipa sensoru pieslēgšana PLC ieejām un iekšējam barošanas avotam dota principiālajā shēmā 4.17. attēlā. PLC ieejām ir jābūt kopējā plusa slēgumā. Kontrollera ieeju kopīgo spaili S0 savieno ar barošanas avota plusa spaili, no šīs spailes padod plusa barošanas spriegumu arī sensoram.

Ja NPN sensors nostrādā (skat. 4.17. att.), tas padod spriegumu 0V(-24 V) uz PLC ieejas otro spaili I0.0. *NPN sensors ieslēdz masu (mīnusu)*.

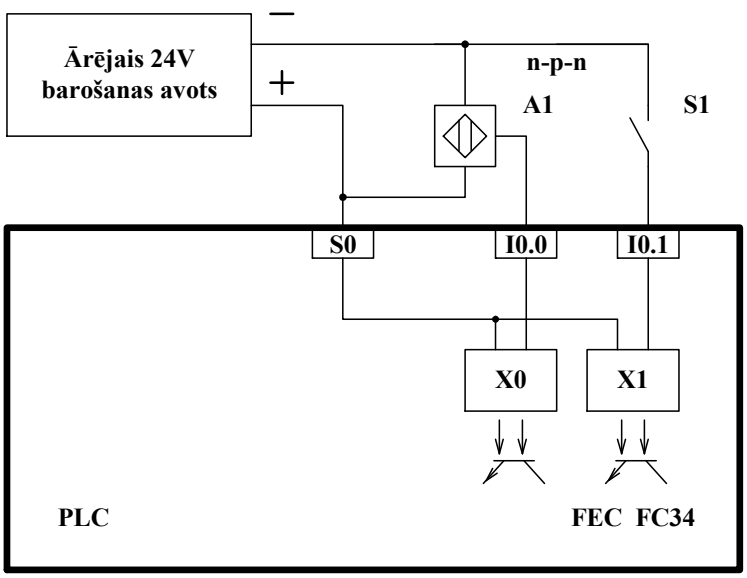


4.16. att. **PNP sensoru pievienošana PLC ar ārēju barošanas avotu**



4.17. att. **NPN sensoru pievienošana PLC ar iekšējo barošanas avotu**

NPN tipa sensoru pieslēgšana PLC ieejām un ārējam barošanas avotam dota principiālajā shēmā 4.18. attēlā. Kontrollera ieejas ir slēgtas kopējā plusa slēgumā.



4.18. att. **NPN sensoru pievienošana PLC ar ārējo barošanas avotu**

Ja NPN sensors A1 nostrādā (skat. 4.18. att.), tas padod spriegumu 0V(-24 V) uz PLC ieejas otro spaili I0.0. *NPN sensors ieslēdz masu (mīnusu)*.

Uzdevums: PLC ieeju pievienošana

1. Pārbaudiet, kādam spriegumam PLC ieejas ir paredzētas.
2. Pārbaudiet, vai PLC ir iekšējais barošanas avots sensoru pieslēgšanai.
3. Ja PLC ir iekšējais barošanas avots, pārbaudiet, kuras spailes ir savstarpēji jāsavieno, lai padotu barošanas strāvas plusu vai mīnusu.
4. Pievienojiet sensoru PLC ieejai un barošanas avotam. Ja PLC nav iekšējā avota, izmantojiet ārējo barošanas avotu.
5. Pārbaudiet ieejas ķēdi saslēdzot kontaktus. Attiecīgās ieejas LED ir jāiedegas.
6. Izvēlieties pareizo tuvinājuma sensora tipu un pievienojiet to PLC.

7. Pievienojiet tuvinājuma sensoru pie PLC ieejas un barošanas avota.
8. Pārbaudiet ķēdes darbību.

4.4. Izpildierīču pievienošana PLC

Automatizētā procesā, kuru vada, piemēram, PLC, sensori kopā ar darbojošajiem izpildmehanismiem apgādā PLC ar informāciju par procesa norisi. Pamatojoties uz šo informāciju un programmu, kas ieprogrammēta PLC atmiņā, PLC aktivizē izpildierīces - relejus, vārstus utt.

Pievienojot izpildierīci, jāņem vērā:

1. Izpildierīces īpatnības (tehniskā specifikācija).
2. PLC izeju tips.
3. Ārējā barošanas avota nepieciešamība.

Izpildierīces tehniskais apraksts dod informāciju par sekojošiem rādītājiem:

- strāvas tipu - līdzstrāva vai maiņstrāva;
- sprieguma vērtību;
- jaudu;
- strāvas stiprumu;
- slodzes tipu;
- ieslēgšanas brīdī nepieciešamo strāvu.

PLC izeju tips:

- releja izeja;
- tranzistora izeja;
- triaka (simistora) izeja.

Ārējais barošanas avots:

- spriegumu un jaudu nosaka viena vai vairākas izpildierīces.

Blokveida PLC var būt vairaki izejas moduļi, kas var būt ļoti dažādi, atkarībā no tā, uz ko izejas attiecas. Tomēr katram modulim parasti ir tikai viena tipa izejas.

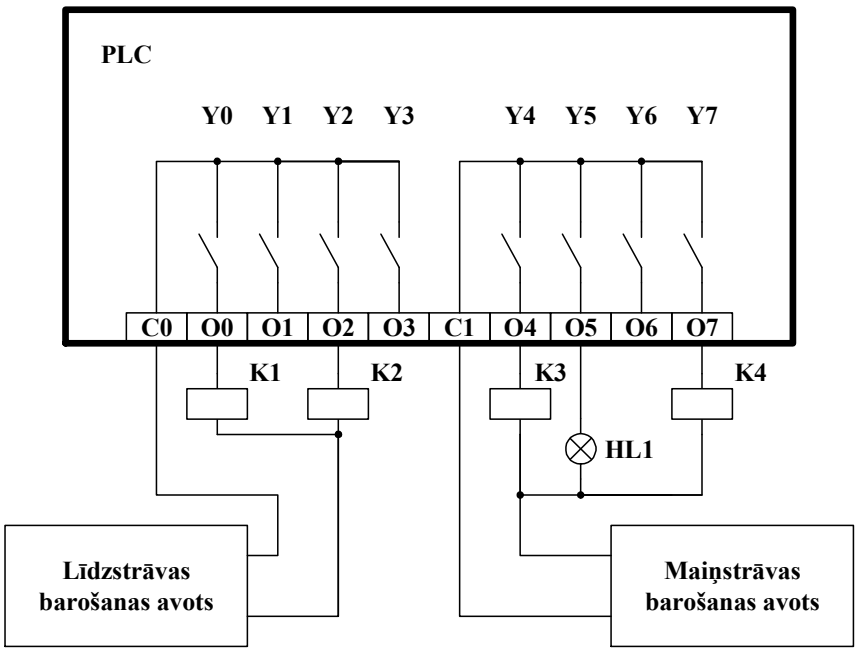
Kompaktajam PLC var būt dažāda tipa izejas, piemēram, FC34 ir gan releju, gan tranzistoru izejas.

PLC releja izejas var strādāt gan ar līdzstrāvu, gan ar maiņstrāvu.

PLC tranzistora izejas, var strādāt tikai ar līdzstrāvu.

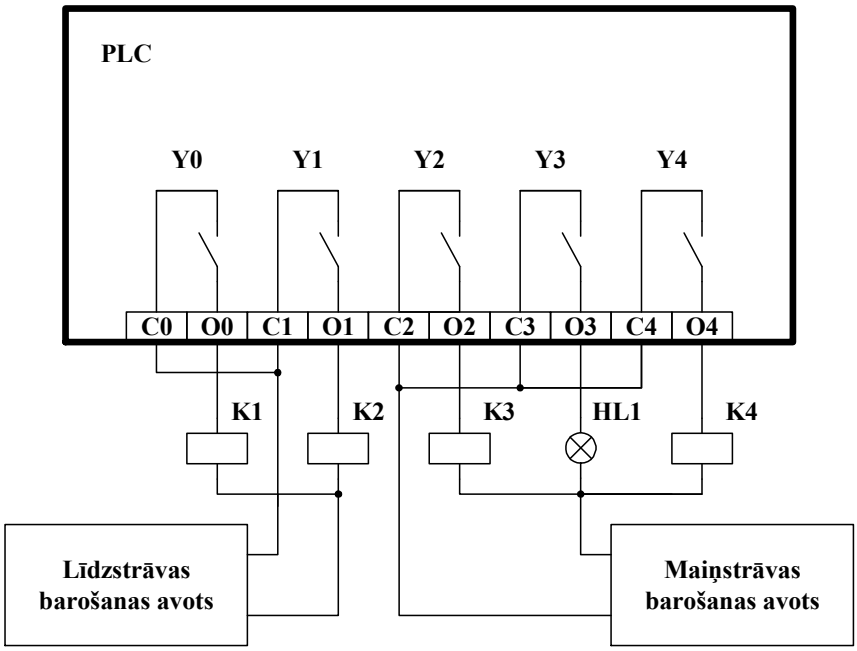
PLC triaka (simistora) izejas, var strādāt tikai ar maiņstrāvu.

PLC izeju savienojošo spaiļu izpildījums var būt dažāds.



4.19. att. **PLC izeju pieslēgums grupā pa četri**

Shēma, kas parāda četru sagrupētu releja tipa izeju slēguma pielietojumu, redzama 4.19. attēlā. Uz katrām četrām izejām nepieciešams ārējais barošanas avots.

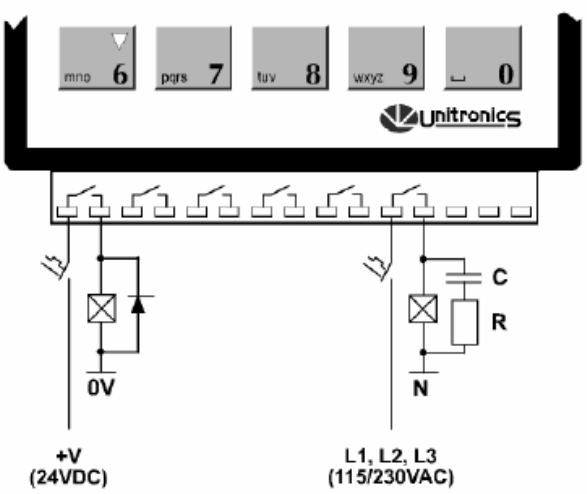


4.20. att. **Releju grupu savienojums ar atsevišķām spailēm katrai izejai**

Shēma, kur katrai izejai (releja izejai) ir divas atsevišķas savienojošās spailes, redzama attēlā 4.20. Katrai izejai ir nepieciešams savs ārējais barošanas avots. Katru izeju var darbināt kā atsevišķu atsaistītu vadības ķēdi.

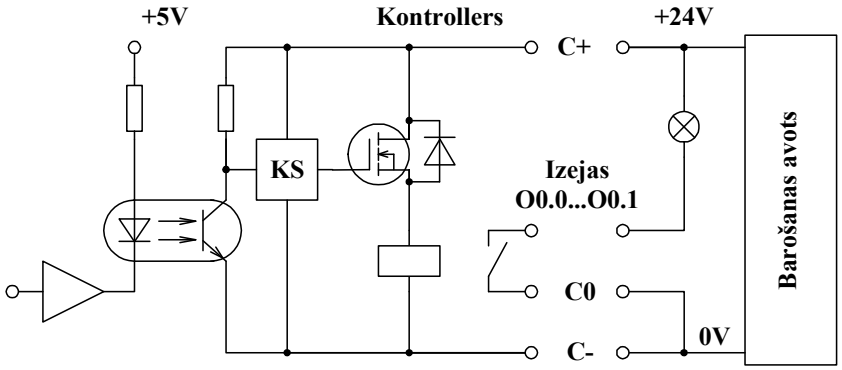
Lai palielinātu releja kontaktu resursu jāizpilda vairāki nosacījumi (skat. 4.21. att):

- Izejas ķēdes ir jāaizsargā pret īsslēgumiem.
- Ja kontakti slēdz līdzstrāvu un tiek slogoti ar induktīvu slodzi, tad paralēli induktivitātei jālieto diode pašindukcijas EDS dzēšanai.
- Ja kontakti slēdz maiņstrāvu paralēli kontaktiem jāslēdz RC ķēde.
- Jāņem vērā, ka pieļaujamā komutācijas strāva induktīvās slodzes slēgšanai ir būtiski mazāka nekā rezistīvai slodzei (piemēram, *Vision* 120 kontrollerim, 5A max rezistīvai, 1A max induktīvai slodzei).



4.21. att. **Izejas ķēdes izmantošana dažādu spriegumu komutācijai**

Arī kontrollerim FC34 ir releju izeja, bet tikai divas O0.0 un O0.1 (skat. 4.22. att.), pārējās ir lauktranzistoru izejas ar vaļēju izteci O0.2 līdz O0.7. Releju izejām ir ierobežota ātrdarbība, to nosaka konkrētā releja parametri.

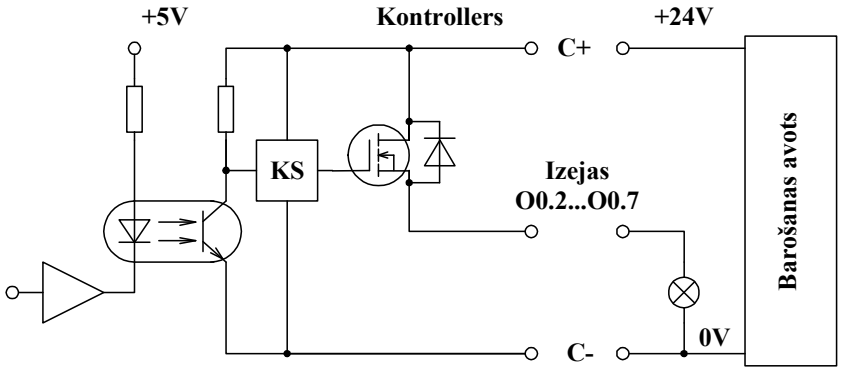


4.22. att. **FC34 kontrollera izejas slēgums ar releja kontaktiem**

Piemēram, FC34 kontrollera releja izeju ātrdarbība ir 25 Hz max pie 5 V sprieguma un 10 mA slodzes. Releju kontaktu resursu būtiski iespaido komutējamā strāva un slodzes veids, rezistīvai slodzei 0,2 A – 1 milions ciklu, 1 A – 0,5 milioni

ciklu, 2 A – 0,3 milioni ciklu, bet induktīvai slodzei 0,2 A – 0,8 milioni ciklu, 1 A – 0,3 milioni ciklu, 2 A – 0,1 milions ciklu, bez slodzes, šiem relejiem paredzēts 20 milionu ciklu. Maksimālā komutācijas strāva 5A/250V maiņstrāva un 5A/30V līdzstrāva.

Tranzistoru izejas ātrdarbība FC34 kontrollerim ir līdz 1 kHz. Nominālā izejas strāva 0,6 A, ja slogo ar lampiņu, tad 5 W. Shēmas izeja ir aizsargāta pret īsslēguma strāvu 4 A, 200 ms.



4.23. att. **FC34 kontrollera izejas slēgums ar vaļēju izteci**

Uzdevums: PLC izeju pievienošana

Pie PLC izejām var pievienot šādus elementus:

- 4 līdzstrāvas relejus ar barošanas spriegumu 24V;
- 2 maiņstrāvas 230V relejus;
- 1 maiņstrāvas 24V zummeru (elektrisks skaņas signāls);
- 1 kontrollampiņu ar barošanas spriegumu 24V.

Konkrētajam PLC ir releju kontaktu izejas.

Pirmkārt, jāiepazīstas ar izpildierīču tehnisko dokumentāciju, lai noskaidrotu:

- barošanas sprieguma tipu;
- barošanas sprieguma vērtību;
- patērēto strāvu;
- palaišanas strāvu;

- slodzes raksturu (aktīva, induktīva).

Tad nepieciešams iepazīties ar PLC izeju specifikāciju, ir jānoskaidro, kam paredzētas izejas, ko pie tām var pieslēgt.

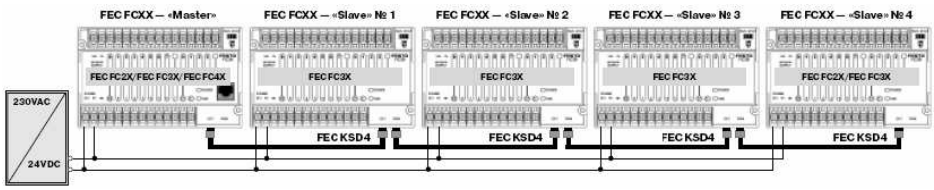
Svarīgākie rādītāji ir:

- maksimālais spriegums;
- maksimālā strāva;
- maksimālais spriegums un strāva pie attiecīgā rakstura slodzes.

Uzzīmējiet shēmu!

4.5. Ieeju un izeju skaita palielināšana

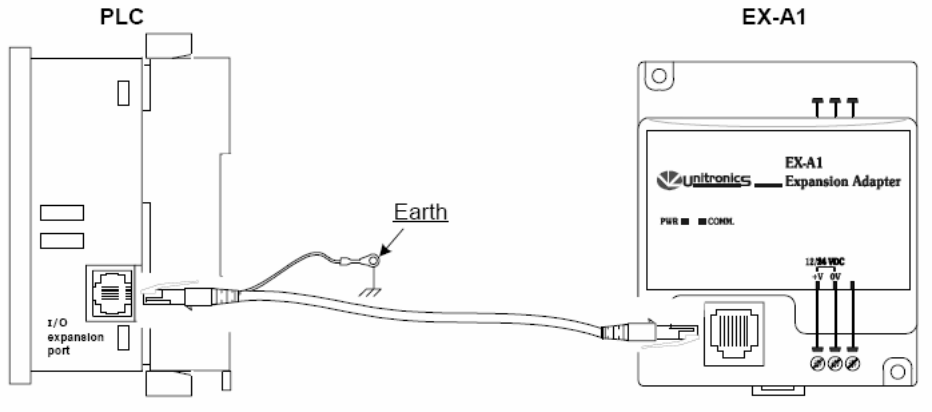
Kompakto FEC FCXX sērijas kontrolleru ieeju un izeju skaitu var palielināt, caur virknes interfeisu papildus pieslēdzot šīs sērijas kontrollerus. Ir galvenais kontrollers – „maser", pārējie ir – „slave", tie ir pakļauti galvenajam. Programmu instalē tikai galvenajā kontrollerī. Ar tīklu komunicē arī tikai galvenais kontrollers (skat. 4.24. att.). FEC (Front End Controller).



4.24. att. **FEC Compact kontrolleru ieeju un izeju skaita palielināšana**

Lai palielinātu ieeju un izeju skaitu Vision 120 kontrollerim ir nepieciešams speciāls paplašināšanas adapters (skat. 4.26. att.), ar speciālu vadu to pieslēdz kontrollerim, pie adaptera var pieslēgt paplašināšanas moduļus (skat. 4.25. att.).

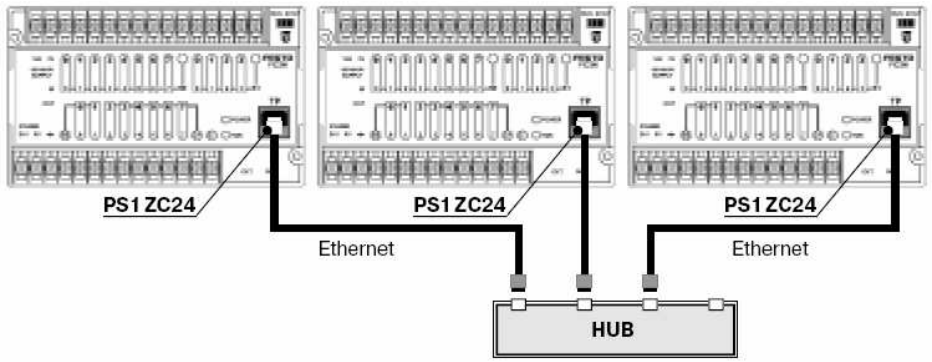
4.25. att. **Ieeju un izeju skaita palielināšana Vision 120 kontrollerim**



4.26. att. **Paplašināšanas bloka pieslēgšana Unitronics kontrollerim**

4.6. Kontrolleru pieslēgšana internetam

Lai kontrolleri pieslēgtu internetam tam ir jāpiešķir IP adrese un jākonfigurē dators komunikācijai ar PLC. Kontrollerus savieno ar LAN tīklu, kā arī var veidot lokālo tīklu, kurā izmanto HUB vai SWITCH tīkla sadalītājus.



4.27. att. **FEC Compact kontrolleru pieslēgšana pie interneta caur HUB**

5. PLC PROGRAMMĒŠANAS VIDE

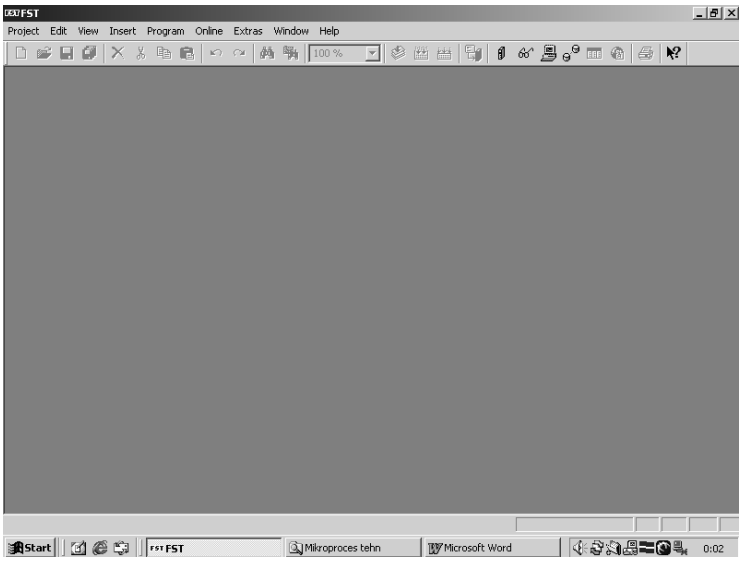
Katrs PLC ražotājs piedāvā savu produktu komplektā ar programmatūru, kas nepieciešama, kontrollera iestatīšanai un programmēšanai. Kontrolleru uzbūve, struktūra un tehniskais risinājums dažādu firmu izstrādājumiem ir atšķirīgs, tāpēc vairumā gadījumos, pievienošanas adapteri un programmatūra ir izstrādāti konkrētajai iekārtai. Vienas firmas dažādu modeļu PLC programmēšanas pievienošanas adapteri arī var būtiski atšķirties. Piemēram, FESTO kompaktajam PLC FC 20 programmēšanu var izpildīt tiešā veidā ar RS232 interfeisa signālu, bet nedaudz jaunāklajam FC34, programmēšanai nepieciešams interfeisa pārveidotājs, kas RS232 signālu pārveido TTL formātā.

Programmēšanas vide iepriekš minētajiem PLC ir viena un tā pati, var izmantot FST programmatūru, kas paredzēta attiecīgo FESTO loģisko kontrolleru sagatavošanai darbam. Šī programmatūra izmantojama tiem kontrolleriem, kuru konfigurācijas informācija un draiveri ir ietverti programmā. Ir svarīgi ar kādām programmēšanas valodām ļauj strādāt sttiecīgā kontrollera programmatūra. Ražotājs mēdz piegādāt programmas versijas, vai programmatūru, kurā ir iespējams programmēt tikai ar kāpņu diagrammām (Ladder diagram), kas bieži vien nav ērtas IT speciālistiem, kuriem parasti ērtāk izmantot teksta valodu (Statement list).

Programmatūra FST ir izmantojama firmas FESTO programmējamo loģisko kontrolleru PLC un industriālo kompjūteru IPC sagatavošanai darbam. Programmas FST versija 4.10.50 nodrošina programmēšanu ar kāpņu diagrammām *Ladder diagrams* un teksta veidā *Statement list*.

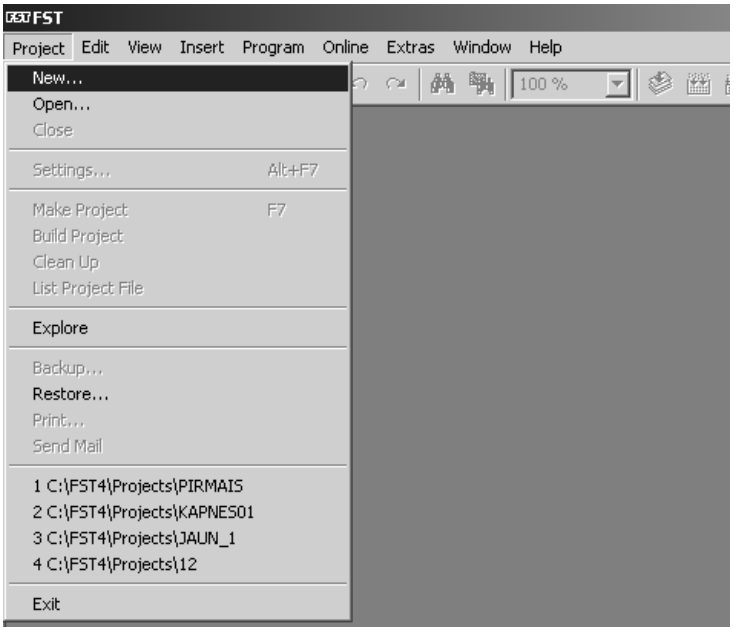
5.1. Programmas atvēršana

Windows programmas loga apakšējā kreisajā stūrī nospiežam starta pogu ***Start*** un caur starta izvēlni ***Programs/Accessories/Festo software/FST4*** iedarbinām programmu *FST4.10.50*, parādās programmas *FST4* darba logs (skat. 5.1. att.).



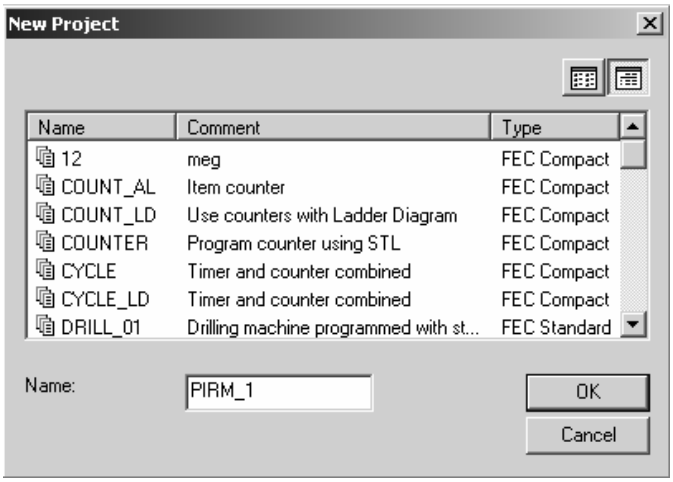
5.1. att. **Programmas FST4 darba logs**

Lai izveidotu jaunu projektu, no programmas komandrindas ņemam izvēlni ***Project***, (skat. 5.2. att.) tad izvēlamies komandu ***New*** (jauns…). Ja vēlamies darbināt iepriekš veidoto projektu, tad to varam paņemt tieši no izvēlnes loga vai ar komandu ***Open***.

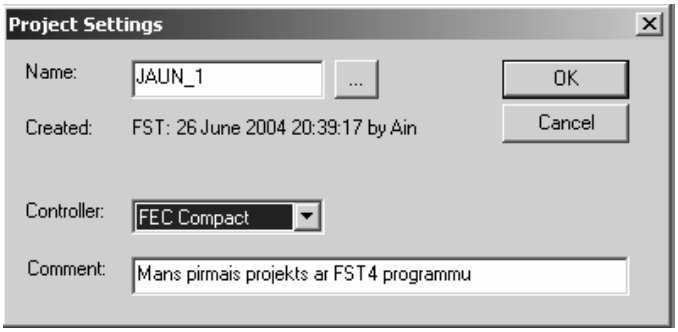


5.2. att. **Jauna projekta atvēršana**

Komanda ***New*** atver projekta uzraudzības logu ***New Project*** (5.3. att.), kurā ir redzami iepriekš izstrādāto projektu nosaukumi, komentāri par projektu un projekta tips. Jaunā projekta nosaukuma ailē ***Name*** ir jautājuma zīme *(?)*, tās vietā rakstām jaunā projekta nosaukumu, piemēram, *JAUN_1*. Projekta nosaukumā nedrīkst būt vairāk par 8 zīmēm.



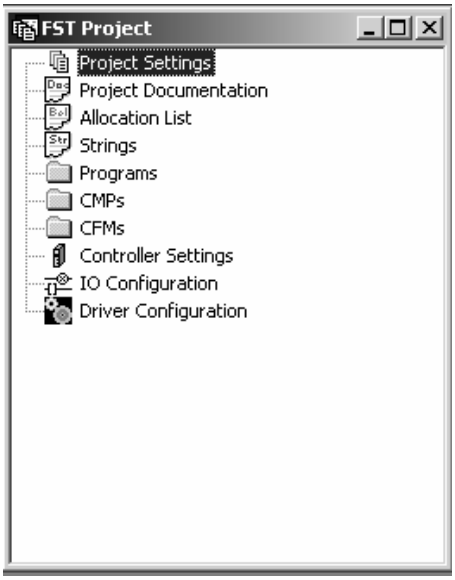
5.3. att. **Jaunā projekta logs**



5.4. att. **Projekta iestatījumu logs**

Apstiprinām projekta nosaukumu ar ***OK***, atveras logs ***Project Settings*** (skat. 5.4. att.). Šai loga ailē ***Name*** ir redzams projekta nosaukums JAUN_1. Ailē ***Controller*** ir jāiestata kontrollera tips, ar kuru tiek veidots šis konkrētais projekts, piemēram, ja

izmantojam kontrolleri FC20 vai FC34, tad ir jāizvēlas *FEC Compact*, ja izmantojam FC660, tad *FEC Standard*, var būt arī citi iestatījumi, tie ir atkarīgi no kontrollera, kuram tiek rakstīta programma.



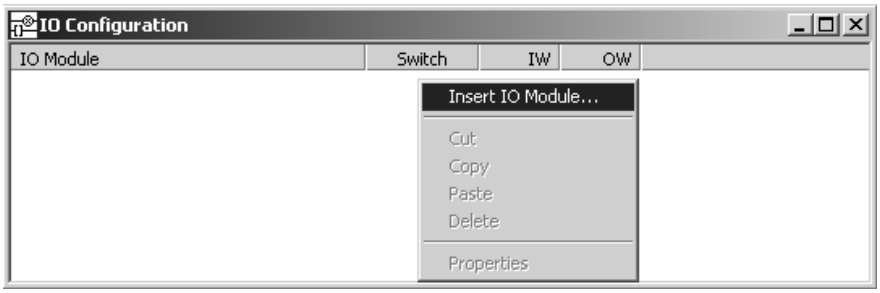
5.5. att. **Projektu logs**

Ailē ***Comment*** varam rakstīt komentārus par projektu, šī ir brīva teksta informācija, kas būs redzama arī projektu atvēršanas logā, kā arī jaunā projekta logā, skatīt 5.3. attēlā. Rakstām tekstu „Mans pirmais projekts ar FST4 programmu". Iepriekš iestatīto apstiprinām ar ***OK***, atveras projektu logs ***FST Project*** (skat. 5.5. att.).

5.2. Ieeju un izeju konfigurācija

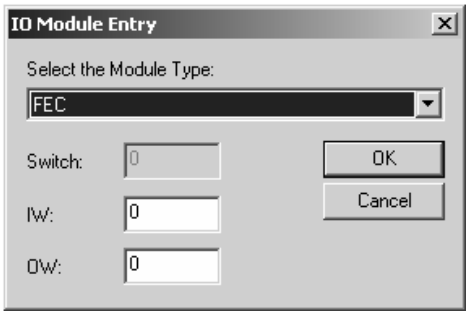
Ir atvērts projektu logs (skat. 5.5. att.) ***FST Project***, bet ja nav atvērts, to var atvērt arī no komandrindas ***View/Project Window***. Sākot jebkuru projektu ir svarīgi kontrollera ieejām un izejām piešķirt adreses. Projektu logā ***FST Project*** izvēlās ***IO Configuration***, atveras ieeju un izeju konfigurācijas logs (skat. 5.6. att.), tas ir tukšs.

Uzspiežot peles labo taustiņu atveras izvēlnes logs, izvēlamies ***Insert IO Module***, atveras ieeju un izeju konfigurācijas modulis ***IO Module Entry*** (skat. 5.7. att.).



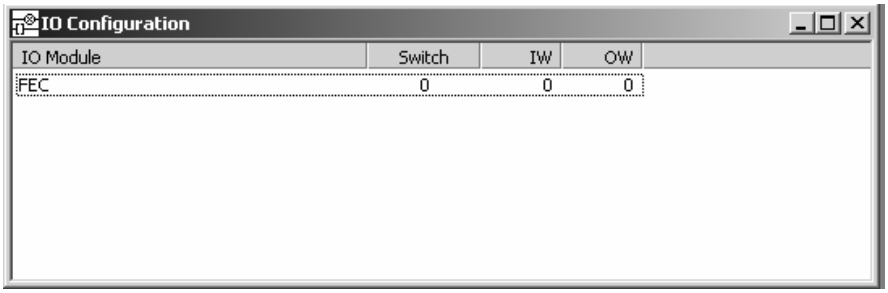
5.6. att. **Ieeju un izeju konfigurācijas logs**

Šai logā ir jāizvēlas vadības moduļa tips, izvēlamies FEC, ja izmantojam FC660, tad izvēlamies to. ***IW*** ir ieejas (input) adrese, ***OW*** ir izejas adrese (output), Ja ***IW*** un ***OW*** atstājam 0, tad absolūtās adreses būs: I0.0 …I0.7 – pirmajai 8 ieeju grupai, I1.0 … I1.7 – otrajai 8 ieeju grupai. Analoģiski adreses būs piešķiramas izejām: O0.0 … O0.7 utt.

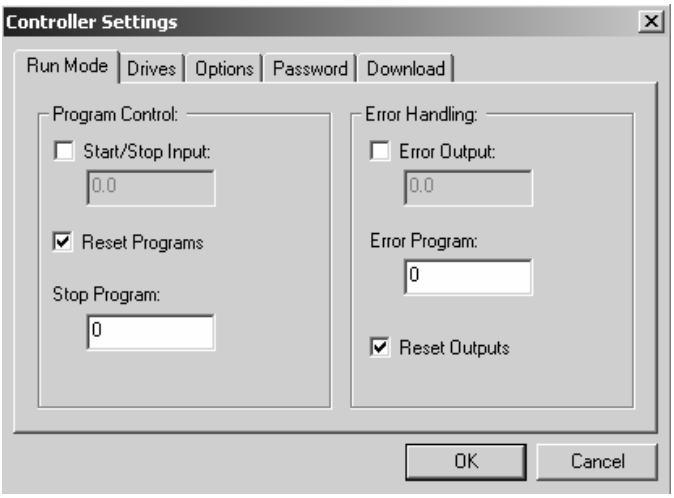


5.7. att. **Ieeju un izeju konfigurācijas modulis**

Ieeju un izeju skaitu nosaka kontrollera konstrukcija. Pēc I/O moduļa (skat. 5.7. att.) iestatījumu apstiprināšanas ar ***OK***, atveras ***IO Configuration*** logs ar attiecīgā kontrollera tipa nosaukumu un ieeju ***IW***, izeju ***OW*** un slēdžu ***Switch*** adresu iestatījumu. Parasti *FEC Standard* un *FEC Compact* tipa kontrolleriem ieeju un izeju adresēm iestata 0.

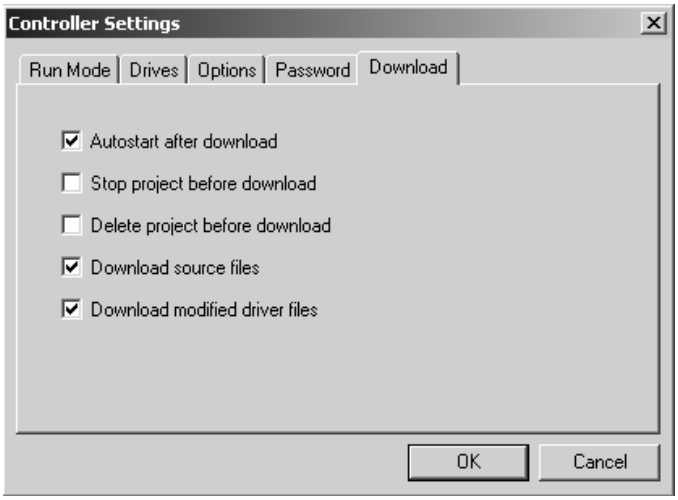


5.8. att. **Ieeju un izeju konfigurācijas logs kontrollerim FEC, (FC34)**



5.9. att. **Kontrollera darbības iestatījumu logs**

Kad ir ievadītas ieeju un izeju adreses, jāiestata kontrollera darba parametri. No projektu loga ***FST Project*** (skat. 5.5. att.) izvēlas ***Controller Settings***, atveras kontrollera iestatījumu logs (skat. 5.9. att.). Izvēlnē ***Run Mode*** iestata kontrollera darbības nosacījumus. Ja ieslēgs ***Start/Stop Input***, tad lai iedarbinātu kontrolleri būs jāpadod vadības signāls uz I0.0 ieeju. Ja aktivēs ***Error Output***, tad kļūdas gadījumā uz izejas O0.0 parādīsies kļūdas signāls.

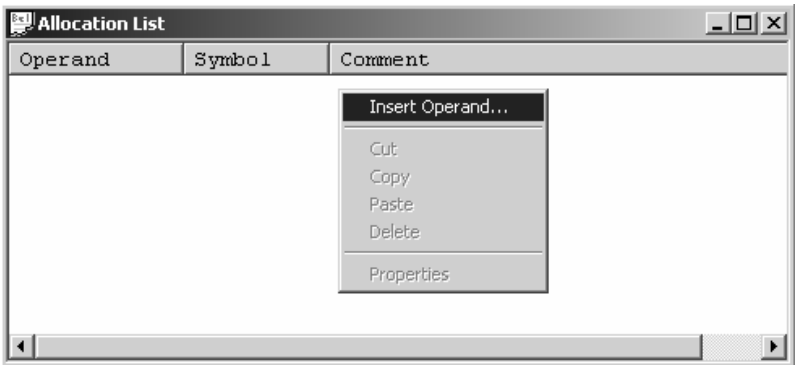


5.10. att. **Programmas lejupielādes logs**

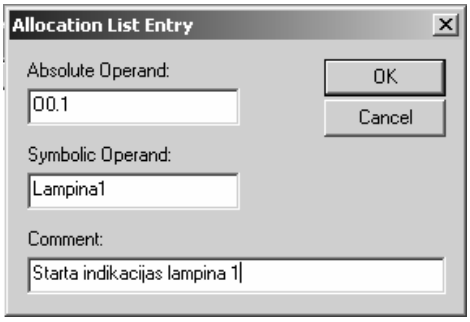
Tad zem ***Controller Settings***, atver programmas lejupielādes logu ***Download*** (skat. 5.10. att.). Šai logā iestata nosacījumus, kuri turpmāk darbosies projekta ielādes procesā. Parasti ieslēdz ***Autostart after download***, kontrolleris no datora saņemto programmu pēc programmas ielādes automātiski to iedarbinās. Ja nebūs šāds iestatījums, programma pēc lejupielādes uzreiz nesāks darboties, tā būs jāiedarbina atsevišķi.

5.3. Operandu iestatīšana

No projektu loga ***FST Project*** (skat. 5.5. att.) izvēlamies un atveram ***Allocation List*** (skat. 5.11. att.).

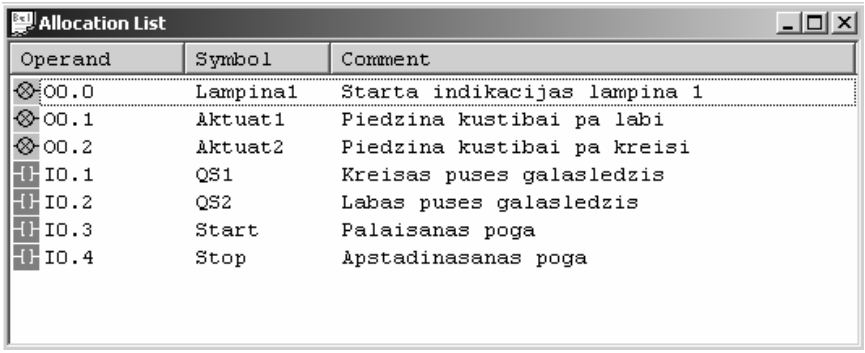


5.11. att. **Operandu lokalizācijas (Allocation List) tabula**



5.12. att. **Operanda iestatīšanas logs**

Piespiežot peles labo taustiņu, atveras izvēlņu logs, ņemam ***Insert Operand*** (ievietot operandu), atveras logs ***Allocation List Entry*** – operanda iestatīšanas logs (skat. 5.12. att.).



5.13. att. **Ieviesto operandu lokalizācijas tabula**

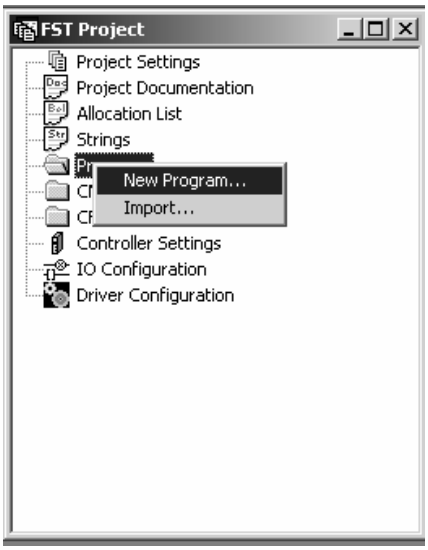
Operandi ir elementi, no kuriem veido programmu, tie ir ieejas, izejas, atmiņas, reģistri, taimeri, skaitītāji utt. Operandu iestatīšanas logā raksta operandu parametrus: laukumā ***Absolute Operand*** – ir jāraksta precīzi, kontrollerim saprotama informācija, tā ir operandu tipi un adreses, piemēram, ieejas – I0.0, izejas O0.0, atmiņas – F0.0, reģistri – R0; laukumā ***Symbolic Operand*** – raksta operandam piešķirto simbolisko vārdu, kas sastāv no 8 zīmēm, piemēram, ieejām – Start, Stop, Sens01, Max1, utt., izejām – Lampina, Varsts, Relejs; laukumā ***Comment*** – raksta komentāru par attiecīgā

elementa funkciju. Ja ierakstīto apstiprina ar ***OK***, tad jaunizveidotais operands tiek ievietots operandu lokalizācijas tabulā ***Allocation List*** (skat. 5.13. att.).

Atkārtoti atverot operandu iestatīšanas logu (skat. 5.12. att.), var izveidot projektam nepieciešamos operandus. Jaunu operandu izveidošanai var izmantot arī citus paņēmienus, viens no tiem ir jaunu operandu ieviešana tieši programmas izveides laikā.

5.4. Programmas izveidošana

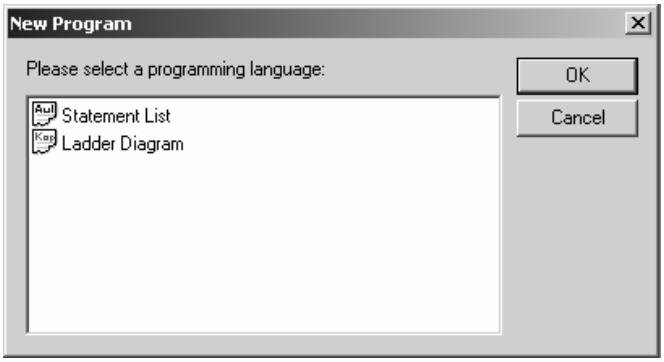
Projekts ir atvērts, operandi izveidoti, var sākt veidot vadības programmas kontrollerim. Projektu logā izvēlamies ***Programs*** un uzklikšķinām ar peles labo taustiņu, atveras logs, izvēlamies ***New Program…*** (skat. 5.14. att.), atveras logs programmēšanas valodas izvēles logs ***New Program*** (skat. 5.15. att.).



5.14. att. **Jaunas programmas izveidošana**

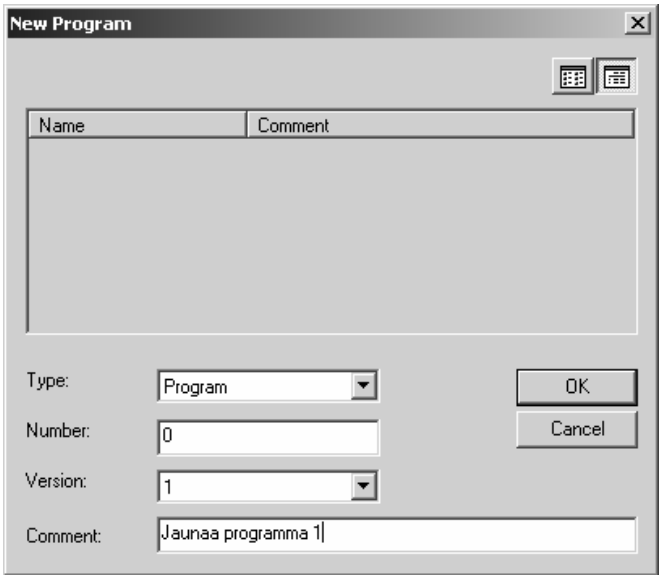
FST programmas 4.10.50 versija piedāvā programmēšanu veikt izmantojot kāpņu diagrammas ***Ladder Diagram***, vai teksta programmēšanu ***Statement List***. *Ladder Diagram* programmēšana ir analoģiska releju shēmu veidošanas filozofijai, tās izmantošana izdevīga agrāk uz releju bāzes projektēto vadības skapju aizstāšanai ar

modernu kontrolleri. Jaunu projektu izveidei ērtāka un mazāk darbietilpīga ir programmēšana lietojot valodu *Statement List*, tā ir izveidota pamatojoties uz plaši pazīstamu programmēšanas valodu principiem. Izvēlamies programmēšanas valodu ***Statement List*** , spiežam pogu ***OK***, atveras jaunās programmas logs (skat. 5.16. att.).



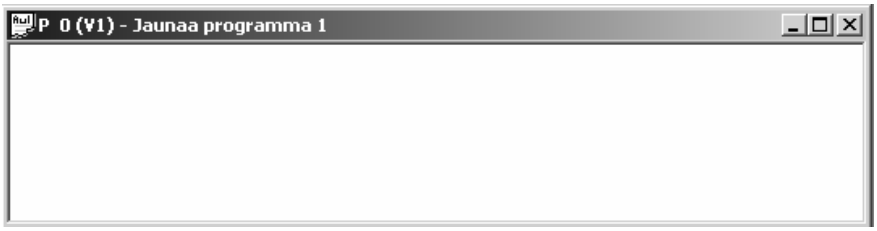
5.15. att. **Programmēšanas valodas izvēles logs**

Vienā projektā var ietilpt vairākas programmas, progammu numerācija jāsāk ar 0, katrai programmai ir vairākas versijas. Logā ***New Program*** (skat. 5.16. att.) norāda programmas numuru ***Number*** - 0, versiju ***Version*** – 1 un uzraksta komentāru par attiecīgo programmu, piemēram, „Jaunā programma 1", to apstiprina ar ***OK***.



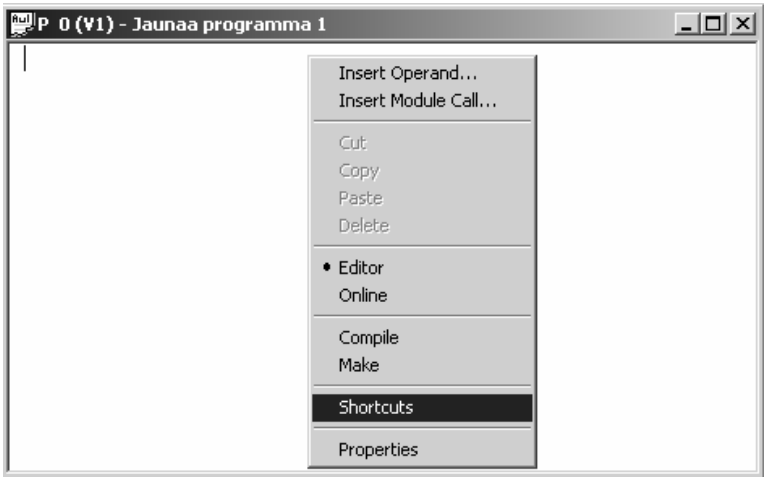
5.16. att. **Jaunās programmas iestatījumu logs**

Rezultātā atveras programmas **P0(V1) – Jaunaa programma 1** programmēšanas logs (skat. 5.17. att.). No programmas nosaukuma var saprast, ka tā ir **0** programmas **1** versija un komentāros dotā informācija ir izlasāma kā programmas nosaukums.



5.17. att. ***Statement List*** **programmēšanas logs**

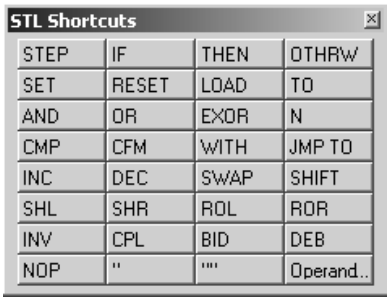
Logs sākumā ir pilnīgi tukšs (skat. 5.17. att.). Ar peles labo pogu uzklikšķinām, atveras programmas vadības izvēlnes logs (skat. 5.18. att.), šai logā ņemam ***Shortcuts***.



5.18. att. **Programmas vadības izvēlne**

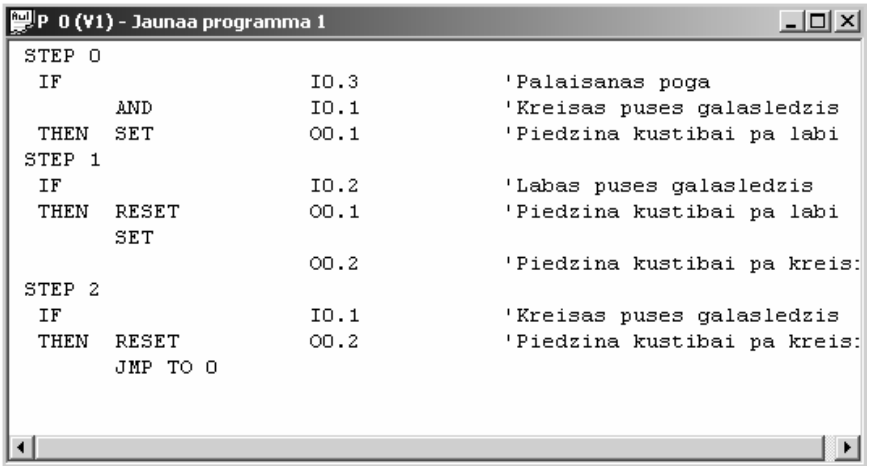
Atveras Statement List (STL) valodas elementu tabula ***STL Shortcuts*** (skat. 5.19. att.). Varam sākt rakstīt programmu. Pieraksta struktūra parasti ir - IF … THEN …, (Ja …Tad), ja kaut kas izpildās, tad kaut kas notiek. Programma var būt pierakstīta vienlaidus tekstā, vai arī pa soļiem - STEP. Ja pieraksta programmu pa soļiem, tad nākošais solis netiek izpildīts, ja nav izpildīts iepriekšējais, protams, ja nav komanda,

kas liek izpildīt nākošo soli. Programmas soļu izpilde notiek momentāni, tāpēc programmā obligāti jāparedz, laika aiztures punkti, vai gala slēdži, ar kuru palīdzību tiek konstatēts izpildierīču stāvoklis.



5.19. att. **STL valodas elementu tabula**

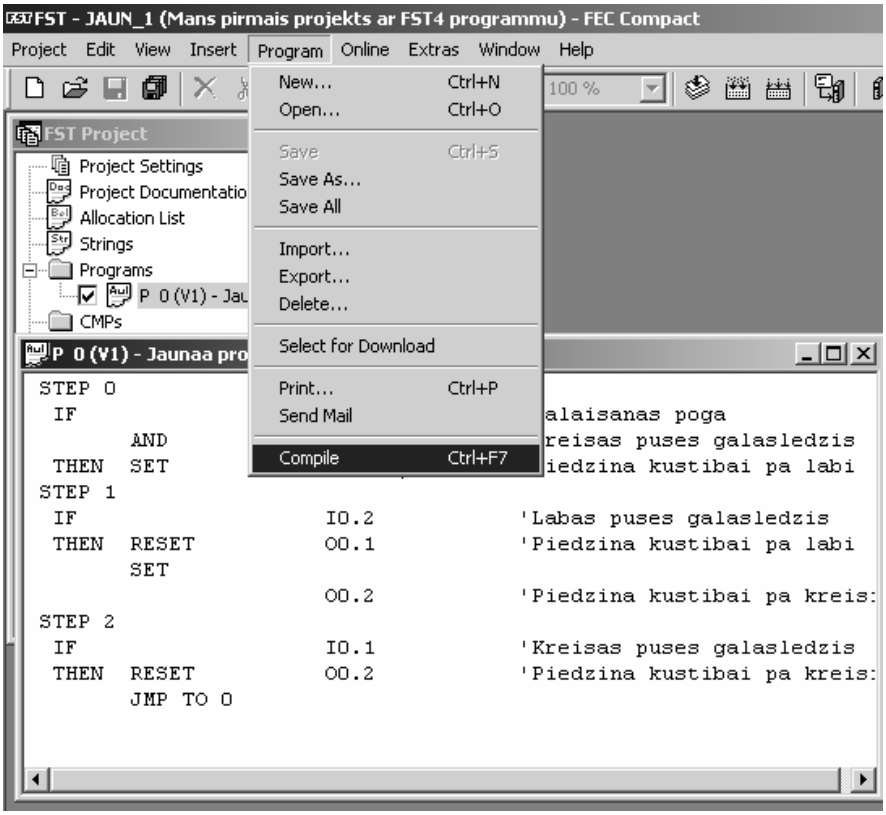
Esam uzrakstījuši programmu iekārtai, piemēram (skat. 5.20. att.), iekārtai ar lineāru turp – atpakaļ kustību. Iekārtai ir izmantota reversīva piedziņa, galaslēdži, stāvokļa konstatēšanai, vadības pogas.



5.20. att. **Programmas redaktora logs**

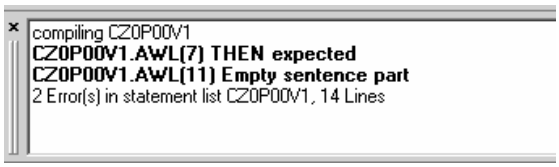
Kad programma ir uzrakstīta, tā ir jāpārvērš kodos, kurus saprot loģiskais kontrollers, ir jāizpilda programmas kompilēšana. FST programmas komandrindā

izvēlamies programmu apstrādes logu ***Program***, ņemam komandu ***Compile*** (skat. 5.21. att.). Momentāni notiek programmas pārbaude.



5.21. att. **Programmas kompilēšana**

Ja kompilēšanas procesā konstatē, ka programmas pierakstā ir pareizrakstības kļūdas, saņemam ziņojumu, kurā ir izklāstīta informācija par kļūdu atrašanās vietu un raksturu un kļūdu skaitu (skat. 5.22. att.).

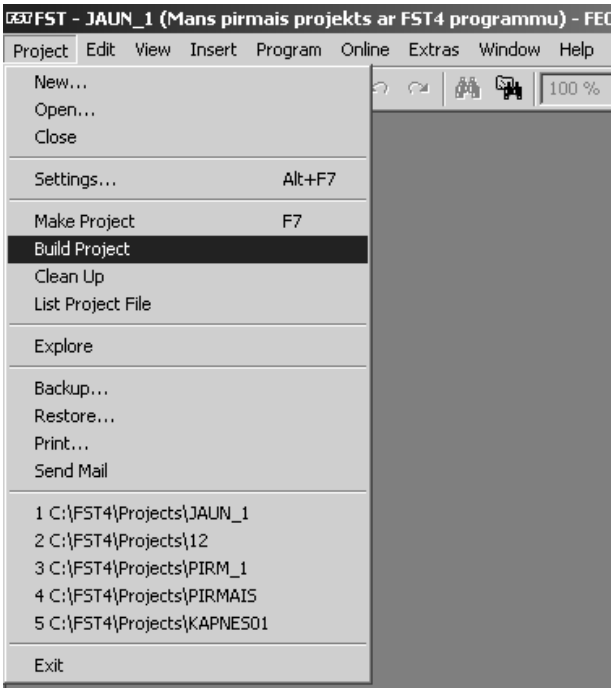


5.22. att. **Kompilēšanas rezultāts - ziņojums par kļūdām programmā**

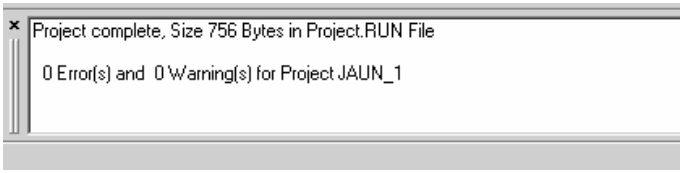
Ja programma ir uzrakstīta pareizi un kompilēšanas procesā dators kļūdas nekonstatē, tad saņemam ziņojumu, ka kļūdu skaits = 0 (skat. 5.23. att.), programmu varam ievietot projektā JAUN_1.

5.23. att. **Kompilēšanas rezultāts - kļūdu nav**

No programmas FST galvenās komandrindas ņemam ***Project/Built Project*** (skat. 5.24. att.) no jauna izveidotā un kompilētā programma **P 0(V1) – Jaunaa programma 1** tiek ievietota projektā JAUN_1. Rezultāti par projektu redzami ziņojumā (skat. 5.25. att.). Kļūdu nav, projekts ir sagatavots ielādei kontrollerī.



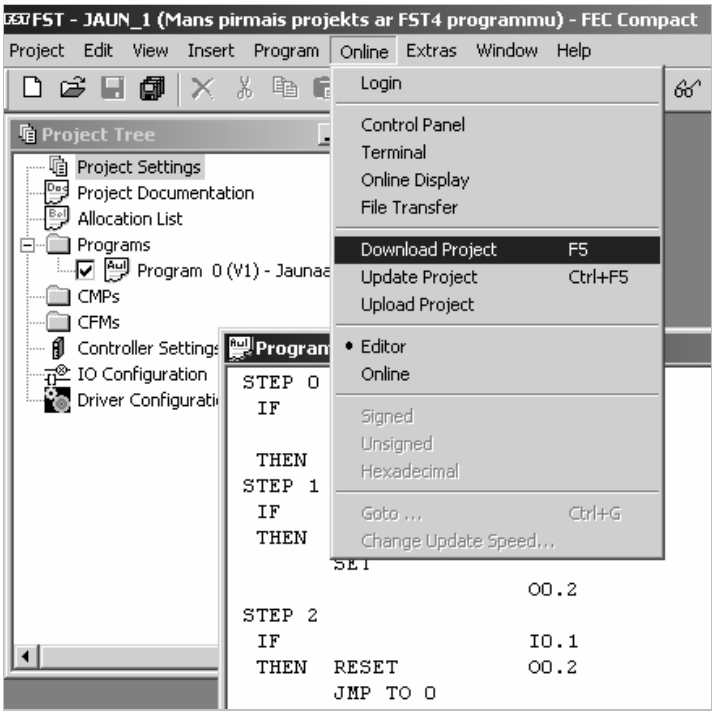
5.24. att. **Programmas ievietošana projektā**



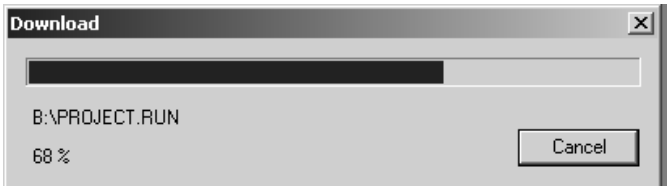
5.25. att. **Projekta pārbaudes rezultāti**

5.5. Projekta lejupielāde kontrollerī

Lai projektu ielādētu kontrollerī, no galvenās komandrindas jāpaņem izvēlne ***Online/Download Project*** (skat. 5.26. att.). Ja kontrollers ir pieslēgts un viss ir kārtībā, notiks projekta ielāde kontrollerī (skat. 5.27. att.), vienīgi var parādīties logi ar jautājumu, vai izdzēst, kontrollerī jau esošo programmu, vai arī brīdinājums, ka tiks izdzēsta esošā programma.



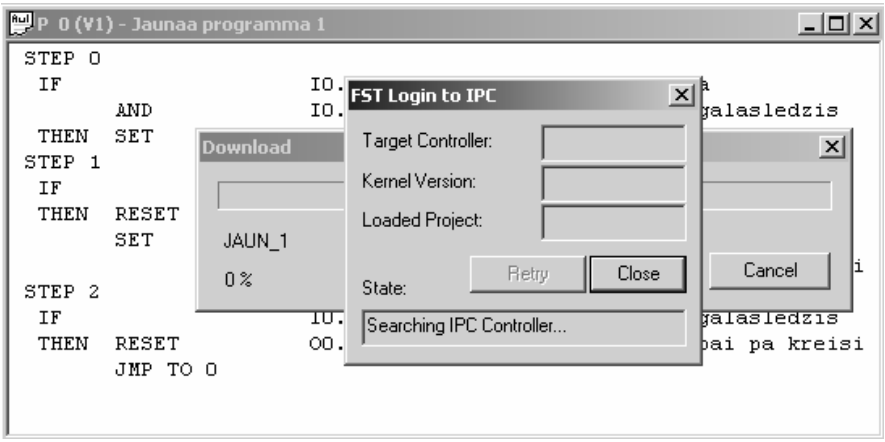
5.26. att. **Projekta nosūtīšana uz kontrolleri**



5.27. att. **Projekta lejupielāde kontrollerī**

Ja šādi brīdinājumi ir no ražošanā izmantota kontrollera, tad pirms turpina programmas ielādi, jābūt pārliecinātam, vai nepazaudēsim derīgu veco programmu. Ja nepieciešams, veco programmu no kontrollera var nokopēt.

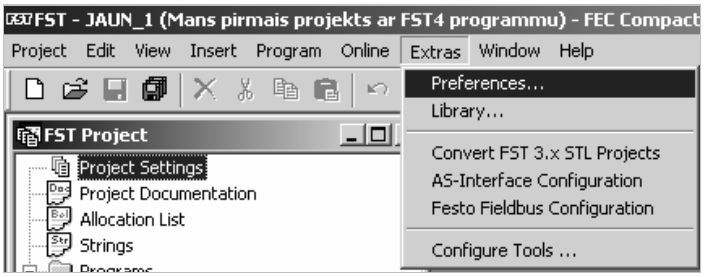
Mēdz būt gadījumi, ka dators nevar atrast kontrolleri, vai nevar to atpazīt, rezultātā uz datora ekrāna ir logs FST Login to IPC ar norādi Searching IPC Controller…(skat. 5.28. att.). Šāda problēma visbiežāk rodas, ja nav pievienots vads, vai nav pareizi izvēlēts datora ports (COM1 vai COM2), nav ieslēgta barošana kontrollerim.



5.28. att. **Kontrollerim nav saites ar datoru**

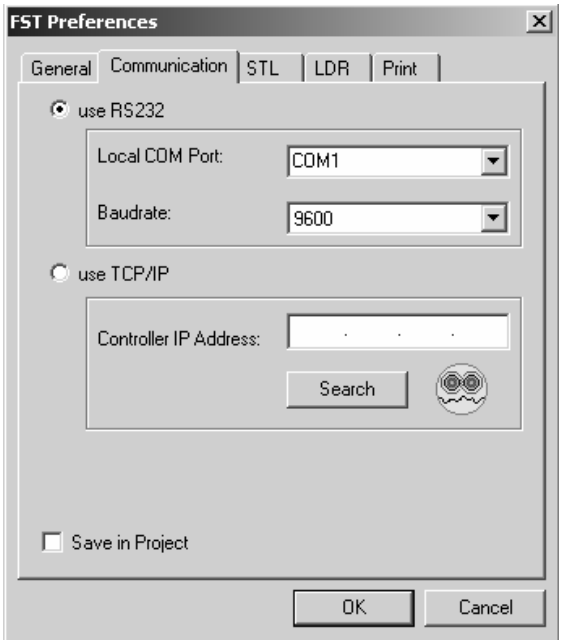
FST programmas iestatījumus var veikt, ja atver no galvenās komandrindas izvēlni ***Extras/Preferences*** (skat. 5.29. att.). Izvēloties ***Preferences*** atvērsies logs ***FST Preferences*** (skat. 5.30. att.), caur šo logu var iestatīt programmēšanas un izdrukas nosacījumus, bet vissvarīgākais, datora komunikācijas veidu ar kontrolleri. Ja

kontrolleri savienojam ar datoru ar vadu caur COM 1 vai COM 2 portu, tad izvēlamies komunikācijas nosacījumu use RS 232, atkarībā no datora konstrukcijas paredzam izmantot vienu no COM portiem, to iestata laukumā ***Local COM Port*** (skat. 5.30. att.).



5.29. att. **FST programmas iestatījumu logs**

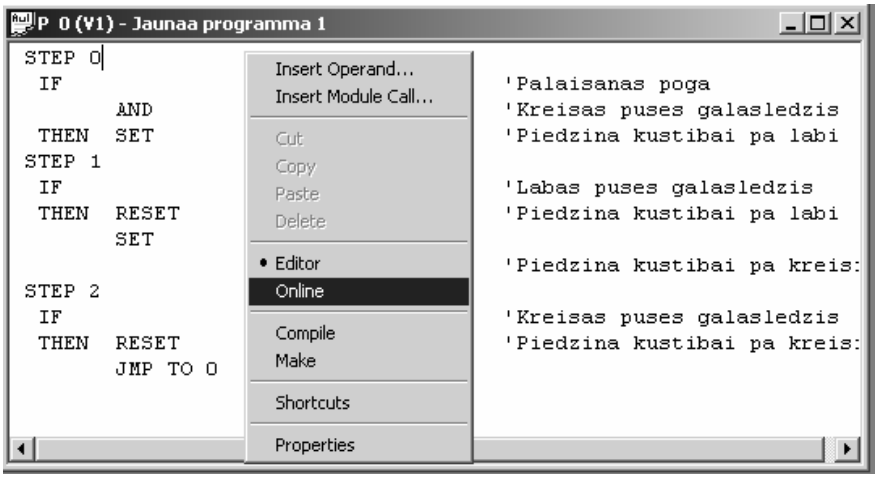
Ja paredzam, ka kontrollers tiks savienots ar datoru caur datortīklu, tad kontrollerim jāpiešķir adrese un tā jāieraksta laukumā ***Controller IP Address*** un jāizvēlas komunikācijas veids ***use TCP/IP***. Ja vēlamies, lai FST programmas iestatījumi saglabātos projektā, izvēlamies ***Save in Project***.



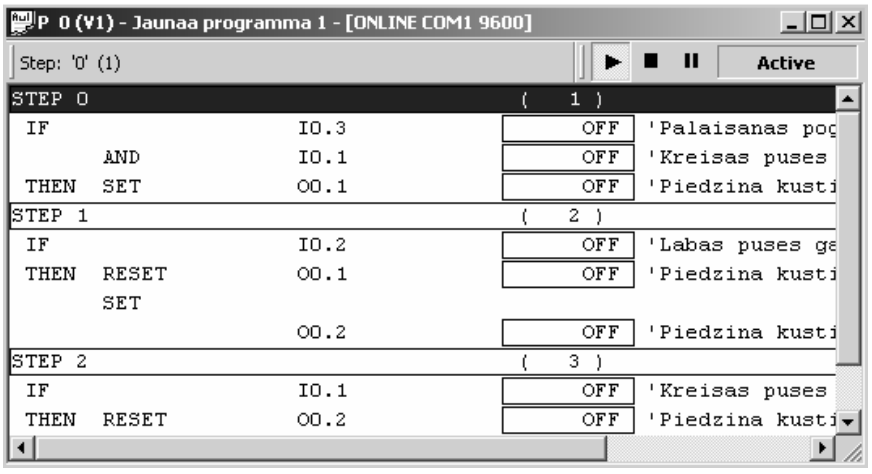
5.30. att. **FST programmas komunikācijas iestatījumi**

5.6. Darbs „On-line” režīmā

Ar peles labo taustiņu uzklikšķinām, atveras programmēšanas izvēlņu logs (skat. 5.31. att.), no tā paņemam ***Online.*** Atveras programmas logs Online režīmā (skat. 5.32. att.) un ir iespēja novērot programmas darbību reālā laikā. Var pārliecināties par kļūmēm programmas izstrādē, konstatēt slēdžu un sensoru nostrādi, meklēt defektus gan programmā, gan iekārtā, kuru programma vada.



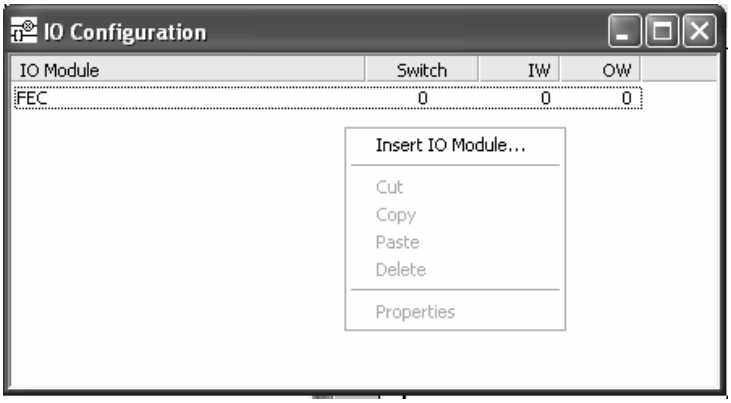
5.31. att. **Programmēšanas izvēlņu logs**



5.32. att. **Programmas logs Online režīmā**

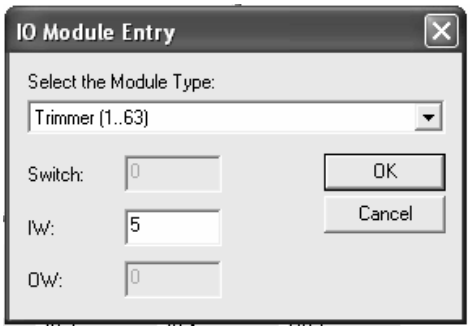
5.7. Trimmera izmantošana

FEC kontrolleriem ir iebūvēts ārējais potenciometrs (Trimmer), kas paredzēts manuālai lieluma iestatīšanai. Piemēram, laika iestatīšanai taimerim, skaitļa iestatīšanai reģistrā utt.

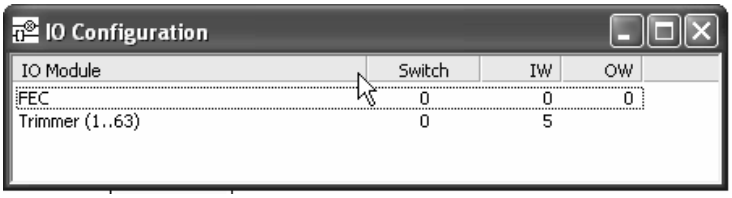


5.33. att. **Ieejas-izejas moduļu iestatīšanas logs**

Lai lietotu potenciometru, ir jāieiet Projekta loga ***FST Project*** ieeju izeju konfigurācijas logā ***I/O Configuration*** un jāiestata Trimmer IW ieejas vārds (skat. 5.33. att.). ***I/O Configuration*** logā uzklikšķina peles labo taustiņu, atveras logs ***Insert I/O module***, ņemam to un iestatām ***Trimmer***. Atveram logu ***I/O Module Entry*** un iestatām Trimmer IW 5 vārdu, parasti vārda numuru iestata no 0 līdz 5 (skat. 5.34. att.).



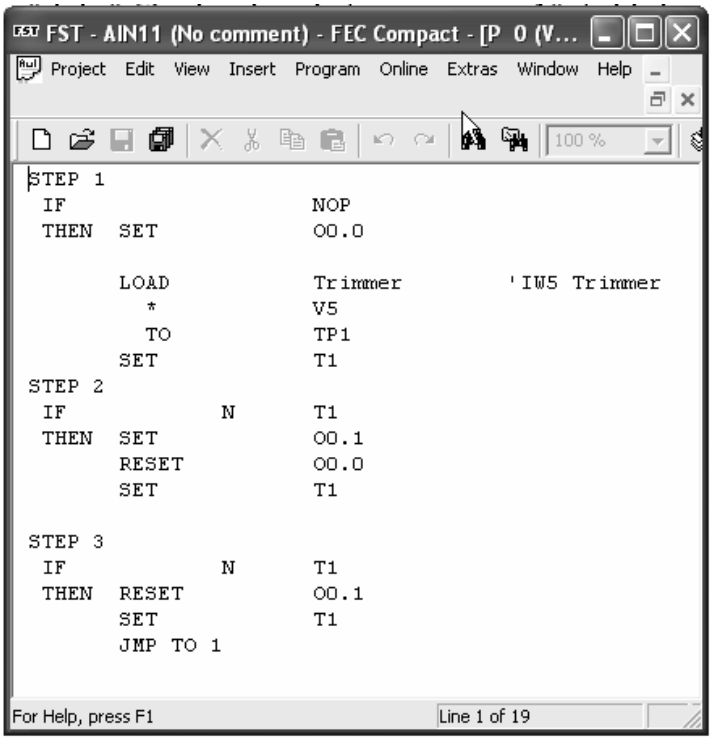
5.34. att. **Ieejas vārda iestatīšana trimmerim**



5.35. att. **Ieeju izeju logs pēc trimmera iestatīšanas**

Ar trimmeri var iestatīt reģistrā lielumu no 0 līdz 63. To var pārbaudīt ieslēdzot iepriekš sagatavotu testa programmu On line režīmā.

Ja to ieraksta taimera iestatīšanas reģistrā TP (Timer Preset), tad laiks mainīsies griežot potenciometru no 0 līdz 0,63 sekundēm. Ja to pareizina ar 10, tad laiks mainīsies no 0 līdz 6,3 sekundēm.

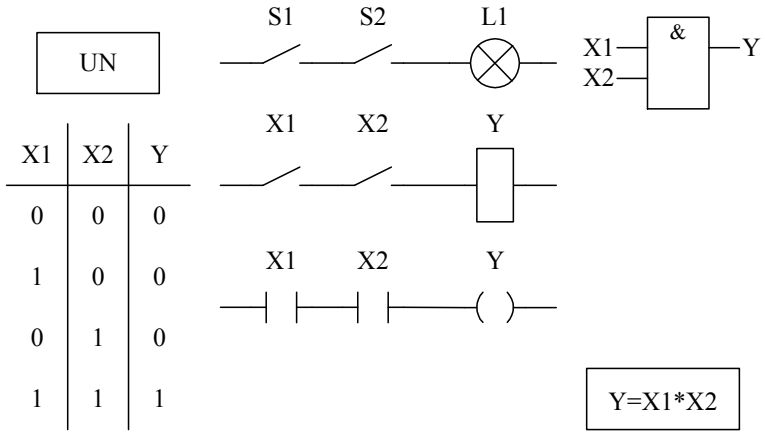


5.36. att. **Trimmera izmantošana taimera vadībai**

6. PLC PROGRAMMĒŠANA

6.1. Loģiskās funkcijas

Visas PLC programmēšanas valodas pamatojas uz loģisko funkciju izmantošanu. Loģiskās pamatfunkcijas ir UN, VAI, NE (AND, OR, NOT), no šīm funkcijām ir izveidotas pārējās UN-NE, VAI-NE, Izslēdzošais VAI utt. Vienu un to pašu loģisko funkciju var parādīt dažādos veidos: ar elektrisko shēmu, loģisko elementu, releju shēmu, gan pēc EN, gan amerikāņu standarta. Loģiskā elementa darbību var aprakstīt ar patiesumvērtību tabulu un loģiskās algebras vienādojumu (skat. 6.1. att.).



6.1. att. **Loģiskā funkcija UN**

Attēlā 6.1. dots loģiskā UN elementa apraksts. Apskatīsim elektrisko shēmu, kas raksturo loģiskā UN elementa darbību. Ja abi slēdži S1 un S2 ir izslēgti, to kontakti ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja slēdzis S1 ir ieslēgts un slēdzis S2 ir izslēgts, S1 kontakti ir saslēgti, bet S2 kontakti ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja slēdzis S1 ir izslēgts un S2 ir ieslēgts, S1 kontakti ir atvērti, S2 kontakti ir saslēgti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja abi slēdži S1 un S2 ir ieslēgti, to kontakti ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Aprakstītais

UN elementa darbības algoritms ir redzams 6.1. attēlā dotajā patiesumvērtību tabulā. Loģisko funkciju UN apraksta loģiskās algebras sakarība,

$$Y = X1 \cdot X2 \tag{6.1}$$

Loģiskajā algebrā izmantojam tikai divas skaitļu vērtības 0 un 1. Loģiskajā algebrā shēmu sintēzē izmantojam šadu aksiomu sistēmu:

- $\overline{0} = 1$, $\overline{1} = 0$, nulles inversā vērtība ir 1, vienīnieka inversā vērtība ir 0;
- $0 \cdot 0 = 0$, $1 + 1 = 1$;
- $1 \cdot 1 = 1$, $0 + 0 = 0$;
- $1 \cdot 0 = 0 \cdot 1 = 0$, $1 + 0 = 0 + 1 = 1$;
- Ja $a = 0$, tad $a \neq 1$, ja $a = 1$, tad $a \neq 0$.

Loģiskās algebras darbības pamatojas uz šīm aksiomām.

Darbības ar nulli:

- $0 \cdot a = 0$;
- $0 + a = a$;
- $0 \cdot a \cdot b \cdot c ... w = 0$.

Darbības ar vienīnieku:

- $1 \cdot a = a$;
- $1 + a = 1$;
- $1 + a + b + c + ... + w = 1$

Pakāpes un reizinājums:

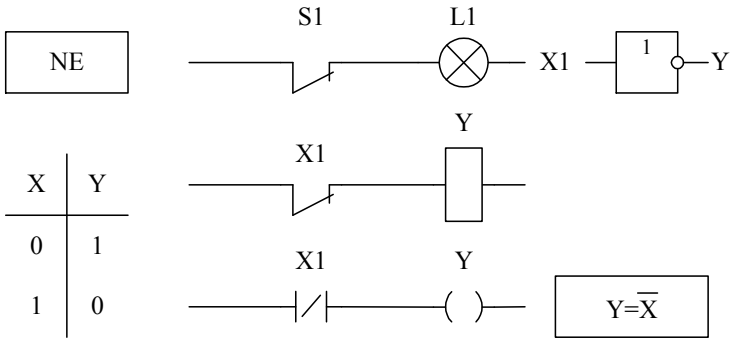
- $a \cdot a = a$;
- $a \cdot a \cdot a ... \cdot a = a^n = a$;
- $a + a = a$;
- $a + a + a + ... + a = n \cdot a = a$.

Attēlā 6.2. dots loģiskā NE elementa apraksts. Apskatīsim elektrisko shēmu, kas raksturo loģiskā NE elementa darbību. Ja slēdzis S1 ir izslēgts, tā kontakti saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Ja slēdzi S1 ieslēdzam, S1 kontakti atslēdzas, ķēde tiek pārtraukta un strāva neplūst, lampiņa L1 nedeg. Aprakstītais NE elementa

darbības algoritms ir redzams 6.2. attēlā dotajā patiesumvērtību tabulā. Loģisko funkciju NE apraksta loģiskās algebras sakarība,

$$Y = \overline{X} \tag{6.2}$$

Tātad ieslēdzot ieeju X1 izeja Y izslēdzas, ieejā ir loģiskais 1, izejā 0 un otrādi.



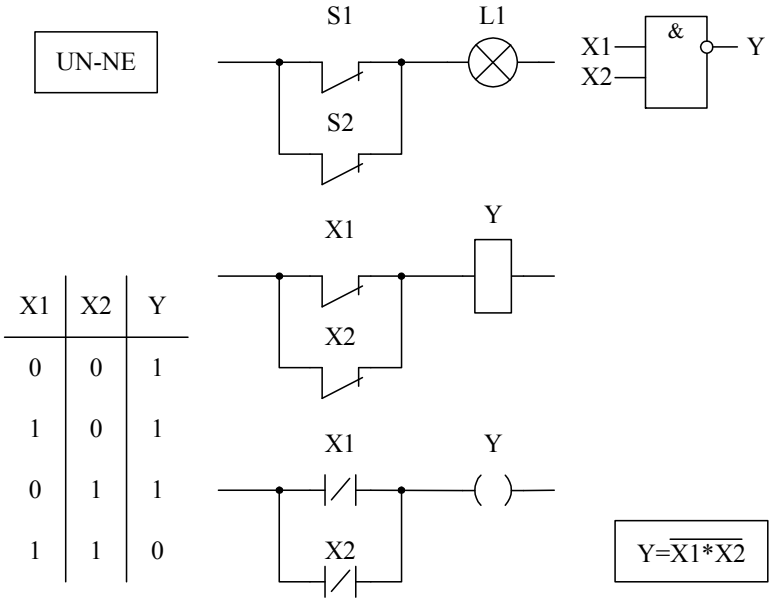
6.2. att. **Loģiskā funkcija NE**

Kombinējot funkcijas UN un NE ir izveidots loģiskais elements UN-NE, tas ir parādīts 6.3. attēlā. Apskatīsim elektrisko shēmu, kas raksturo loģiskā UN-NE elementa darbību. Ja abi slēdži S1 un S2 ir izslēgti, to kontakti ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Ja slēdzis S1 ir ieslēgts un slēdzis S2 ir izslēgts, S1 kontakti ir atvērti, bet S2 kontakti ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Ja slēdzis S1 ir izslēgts un S2 ir ieslēgts, S1 kontakti ir saslēgti, S2 kontakti ir atvērti, ķēdē plūst strāva, lampiņa L1 deg. Ja abi slēdži S1 un S2 ir ieslēgti, to kontakti ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg.

Aprakstītais UN-NE elementa darbības algoritms ir redzams 6.3. attēlā dotajā patiesumvērtību tabulā. Loģisko funkciju UN-NE apraksta loģiskās algebras sakarība,

$$Y = \overline{X1 \cdot X2} \tag{6.3}$$

Tātad ieslēdzot abas ieejas X1 un X2, izeja Y izslēdzas, ja uz abām ieejām ir loģiskais 1, izejā ir 0, visos citos gadījumos izejā ir 1.



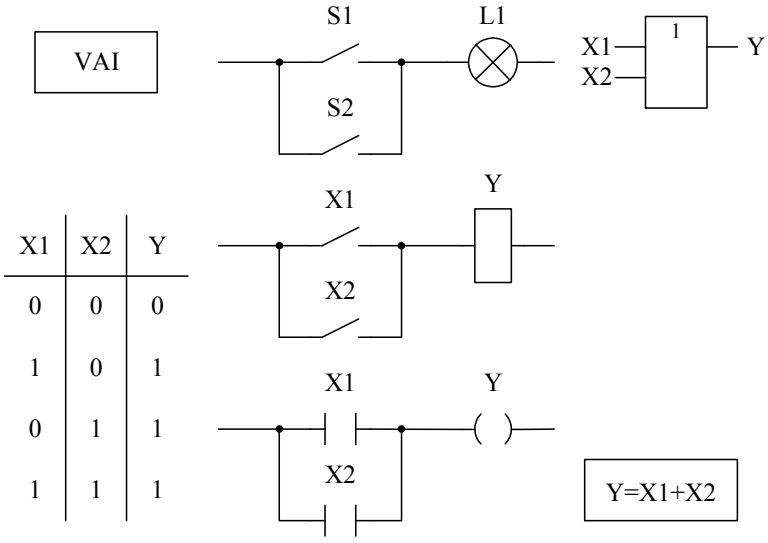
X1	X2	Y
0	0	1
1	0	1
0	1	1
1	1	0

6.3. att. **Loģiskā funkcija UN-NE**

Shēmas, kas veido loģisko elementu VAI ir dotas 6.4. attēlā. Apskatīsim elektrisko shēmu, kas raksturo loģiskā VAI elementa darbību. Ja abi slēdži S1 un S2 ir izslēgti, to kontakti ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja slēdzis S1 ir ieslēgts un slēdzis S2 ir izslēgts, S1 kontakti ir saslēgti, bet S2 kontakti ir atvērti, ķēdē plūst strāva caur S1 kontaktiem un lampiņa L1 deg. Ja slēdzis S1 ir izslēgts un S2 ir ieslēgts, S1 kontakti ir atvērti, S2 kontakti ir saslēgti, ķēdē plūst strāva caur S2 slēdža kontaktiem, lampiņa L1 deg. Ja abi slēdži S1 un S2 ir ieslēgti, to kontakti ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Aprakstītais VAI elementa darbības algoritms ir redzams 6.4. attēlā dotajā patiesumvērtību tabulā. Loģisko funkciju VAI apraksta loģiskās algebras sakarība,

$$Y = X1 + X2 \tag{6.4}$$

Tātad ieslēdzot jebkuru no ieejām X1 un X2, izeja Y ieslēdzas, ja uz vienas vai abām ieejām ir loģiskais 1, izejā ir 1, ja uz abām ieejām ir 0, tad izejā ir 0.



X1	X2	Y
0	0	0
1	0	1
0	1	1
1	1	1

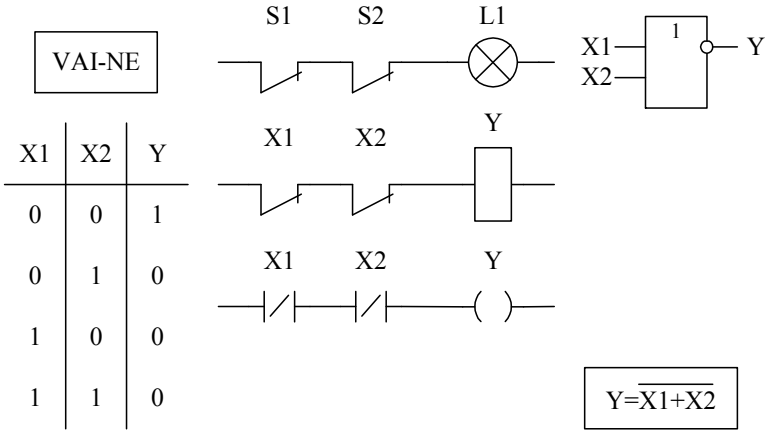
6.4. att. **Loģiskā funkcija VAI**

Kombinējot funkcijas VAI un NE ir izveidots loģiskais elements VAI-NE, tas ir parādīts 6.5. attēlā. Apskatīsim elektrisko shēmu, kas raksturo loģiskā VAI-NE elementa darbību. Ja abi slēdži S1 un S2 ir izslēgti, to kontakti ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Ja slēdzis S1 ir izslēgts un slēdzis S2 ir ieslēgts, S1 kontakti ir saslēgti, bet S2 kontakti ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja slēdzis S1 ir ieslēgts un S2 ir izslēgts, S1 kontakti ir atvērti, S2 kontakti ir saslēgti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja abi slēdži S1 un S2 ir ieslēgti, to kontakti ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg.

Aprakstītais VAI-NE elementa darbības algoritms ir redzams 6.5. attēlā dotajā patiesumvērtību tabulā. Loģisko funkciju VAI-NE apraksta loģiskās algebras sakarība,

$$Y = \overline{X1 + X2} \tag{6.5}$$

Tātad ieslēdzot jebkuru no ieejām X1 un X2, izeja Y izslēdzas, ja uz abām ieejām ir loģiskā 0, izejā ir 1, visos citos gadījumos izejā ir 0.

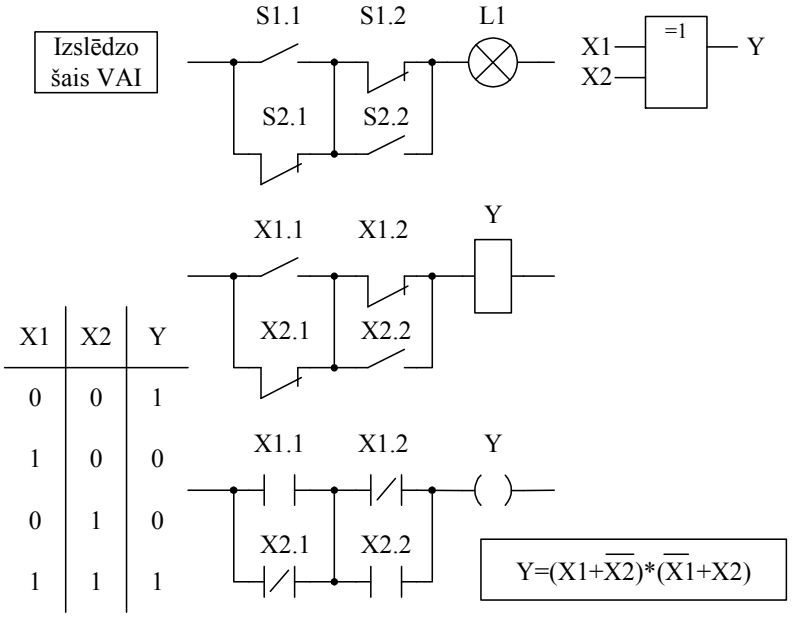


6.5. att. **Loģiskā funkcija VAI-NE**

Nedaudz sarežģītāka funkcija ir Izslēdzošais-VAI (XOR), to sauc arī par ANTIVALENCES funkciju, tās shēma ir parādīts 6.6. attēlā. Apskatīsim elektrisko shēmu, kas raksturo loģiskā Izslēdzošā-VAI elementa darbību. Ja abi slēdži S1 un S2 ir izslēgti, to kontakti S1.2 un S2.1, ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Ja slēdzis S1 ir izslēgts un slēdzis S2 ir ieslēgts, S2 slēdža kontakti S2.1 ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja slēdzis S1 ir ieslēgts un S2 ir izslēgts, S1 kontakti S1.2 ir atvērti, ķēdē strāva neplūst, lampiņa L1 nedeg. Ja abi slēdži S1 un S2 ir ieslēgti, to kontakti S1.1 un S2.2 ir saslēgti, ķēdē plūst strāva, lampiņa L1 deg. Aprakstītais Izslēdzošā-VAI elementa darbības algoritms ir redzams 6.6. attēlā dotajā patiesumvērtību tabulā. Loģisko funkciju Izslēdzošais-VAI apraksta loģiskās algebras sakarība,

$$Y = (X1 + \overline{X}2) \cdot (\overline{X}1 + X2) \tag{6.6}$$

Tātad izeja Y izslēdzas tikai tad, ja uz abām ieejām X1 un X2 ir vienāds statuss, ja uz abām ieejām ir loģiskā 0 vai loģiskais 1, tad izejā ir 1, visos citos gadījumos izejā ir 0.



X1	X2	Y
0	0	1
1	0	0
0	1	0
1	1	1

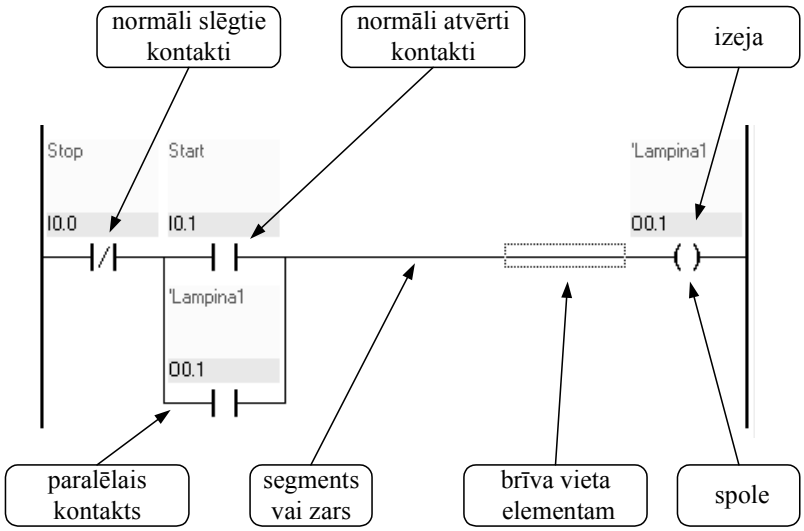
6.6. att. **Loģiskā funkcija Izslēdzošais-VAI**

6.2. Kāpņu diagrammu valoda

Kāpņu diagramma (Ladder Diagram) ir pirmā PLC programmēšanas valoda. Tā ir grafiskā valoda, kuru izveidoja, lai ar PLC aizstātu uz releju bāzes veidotās elektriskās vadības sistēmas. Releju vadības sistēmās izmantoja simbolisko shēmu valodu – kāpņu diagrammu. Kāpņu diagramma sastāv no simbolu virknes, kas savienoti ar līnijām, tādā veidā shematiski attēlojot vadus pa kuriem plūst elektriskā strāva. Kāpņu diagrammu veido divu veidu elementi: slēdža (ieejas) elementi – *kontakti* un slodzes (izejas) elementi – *spoles*, tie saslēgti virknē un veido ķēdes segmentus vai zarus, kas pieslēgti pie enerģijas avota divām kopnēm, veidojot diagrammas pakāpienus.

Kāpņu diagrammu pamatelementi

Rakstot programmu kāpņu diagrammas veidā, tiek izmantotas grafiskās komponentes, kuras savieno loģiskā secībā (skat. 6.7. att.).



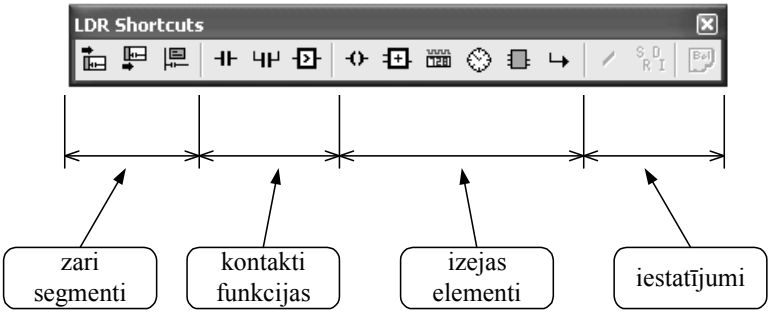
6.7. att. **Kāpņu diagrammas segmenta uzbūve**

Programmā ir izmantoti šādi pamatelementi (skat. 6.7. att.):

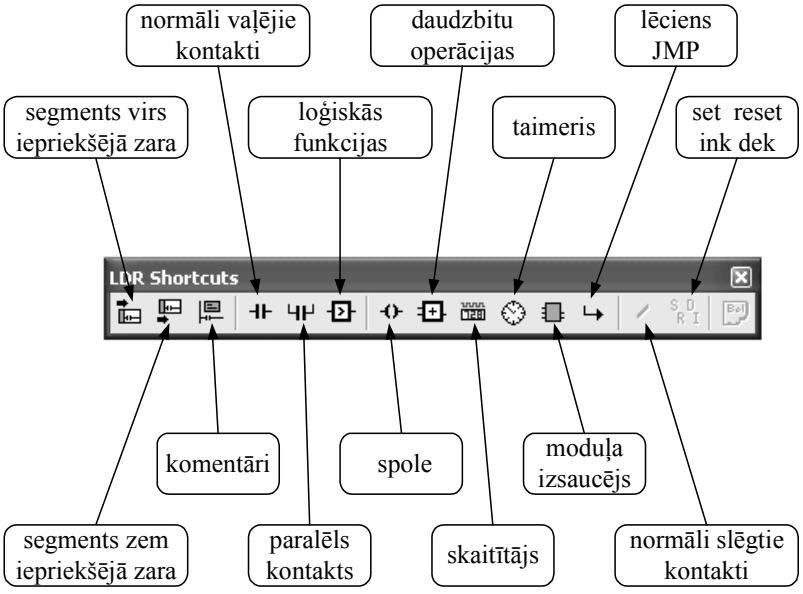
- **Kontakti -** katrs kontakts attēlo slēdzi, caur kuru var plūst strāva, ja kontakti ir saslēgtā stāvoklī;
- **Spoles** - katra spole attēlo releju, kuru ieslēdz caur to caurplūstošā strāva;
- **Elementi** - katrs elements attēlo vienu funkciju, kura tiek realizēta, ja caur to plūst strāva;
- **Segmenti** - katrs segments, vai zars veido noslēgtu strāvas ķēdi. Strāva plūst no kreisās puses kopnes caur saslēgtiem kontaktiem uz spolēm vai elementiem, kuri rezultatā tiek aktivizēti.

Šajā nodaļā kā piemērs aprakstītā LD programmēšanas valoda, kas ir izmantota FST programmatūrā un paredzēta FESTO kontrolleru programmēšanai. LD valodu labprāt izvēlās izmantot elektriķi, kam parasti ir nepieciešamās speciālās zināšanas elektrisko shēmu lasīšanā un projektēšanā. Lai varētu lietot LD valodu, nepieciešams zināt programmas pieraksta grafiskos apzīmējumus, strādājot ar programmu FST, tos izvēlas no loga ***LDR Shortcuts*** (skat. 6.9. att.). Šai logā apzīmējumu ikonas ir

sagrupētas pēc veicamajām funkcijām: darbības ar zariem un komentāri, kontakti un funkcijas, izejas elementi, iestatījumi (skat. 6.8. att.).



6.8. att. **Kāpņu diagrammu elementu loga konfigurācija**



6.9. att. **FST kāpņu diagrammu elementu logs**

Kontakti

Ir divu veidu kontakti, normāli atvērti (NO - normal open) un normāli saslēgtie (NC - normal close) (skat. 6.7. att.). Programmā kontaktus var darbināt:

- kontrollera ieejas I0.0, I0.1, I0.2 utt.;
- kontrollera izejas O0.0, O0.1, O0.2 utt.;
- taimeri T0, T1, T2 utt.;
- skaitītāji C0, C1, C2 utt.;
- flash atmiņas F0.0, F0.1, F0.2 utt.

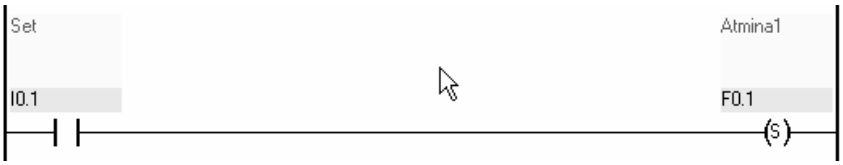
Spole

Ar spoli saprotam izeju, vai slodzi. Programmā spole var izpildīt dažādas funkcijas:

- kontrollera izeja O0.0, O0.1, O0.2 utt.;
- taimera T0, T1, T2... ieslēgšana S (set) un izslēgšana R (reset);
- skaitītāja C0, C1, C2... ieslēgšana S (set), izslēgšana R (reset), pieskaitīšana I (increment) un atskaitīšana D (decrement);
- flash atmiņas F0.0, F0.1, F0.2... ieslēgšana S (set) un izslēgšana R (reset).

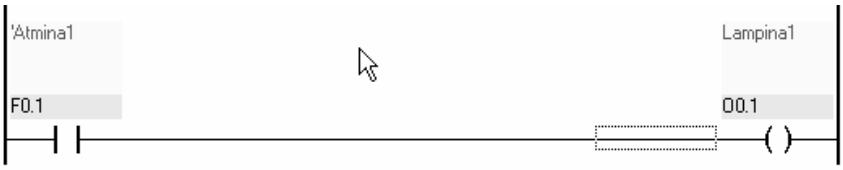
Flash atmiņa

Flash atmiņa ir viena bita atmiņa, kuru izmantojam, ja nepieciešams fiksēt kādu notikumu. Viena bita atmiņu definē un izmanto kā spoli. Atmiņas statuss ir 0 vai 1, izslēgta vai ieslēgta. Flash atmiņas absolūtais operands FST programmēšanas vidē ir F0.0, F0.1, F0.2 utt. Ja atmiņu ieslēdzam (skat. 6.10. att.), tā paliek ieslēgta, kamēr to izslēdz (skat. 6.12. att.).



6.10. att. **Flash atmiņas ieslēgšana**

Ja uz ieeju I0.1 padodam „1”, vai „1” impulsu, viena bita flash atmiņā F0.1 ieraksta 1, atmiņas F0.1 statuss ir 1 (skat. 6.10. att.), par to liecina izeja O0.1, kuras statuss ir 1, izejai pieslēgtā lampiņa deg (skat. 6.11. att.).



6.11. att. **Flash atmiņas izmantošana izejas ieslēgšanai**

6.12. att. **Flash atmiņas izslēgšana**

Jebkurā momentā flash atmiņu var izslēgt, ja ir izveidots elements RESET F0.1 **(**R**)**. Ja uz ieeju I0.2 padodam „1", tad flash atmiņa F0.1 momentāni tiek izslēgta un rezultātā F0.0 statuss ir 0 (skat. 6.12. att.).

Taimeris

Taimeris var skaitīt laiku, ieslēgt un izslēgt ķēdi, formēt impulsa platumu, nodrošināt laika kavējumu starp izpildāmajām operācijām utt.

Katram taimerim ir adrese **T0; T1; T2; T3 … Tnn**.

TPnn (timer preset) skaitlis, uz kuru skaita taimeris, laiks līdz apstāšanās.

TWnn (timer word) vārds, taimera momentānā vērtība *.

Tnn Taimera loģiskais statuss var būt 0 vai 1.

T=1 laiks nav beidzies, taimeris ieslēgts;

T=0 laiks ir beidzies, vai arī nav sākts skaitīt, taimeris izslēgts.

******Taimeris skaita laiku no iestatītā lieluma TPnn uz nulli, kad sasniegta nulle, izslēdzas. Tas jāņem vērā, izmantojot taimera momentāno vērtību TWnn.*

Lai taimeri izmantotu:

- taimeris jādefinē – jāpiešķir nosaukums un jāuzdod sākuma iestatījumi;
- taimeris jāieslēdz – jāpadod spriegums, vai jādod komanda SET;
- jāliek kaut ko darīt – taimerim jāliek ieslēgt, izslēgt citas ķēdes.

Taimera programmēšana

Ja uz I0.1 ieeju padodam „1", vai iedodam pozitīvu „1" impulsu, sāk darboties taimeris T1. Taimeris ieslēgtā stāvoklī ir 5 sekundes un tad izslēdzas. V500 nozīmē Value = 500 sekundes simtdaļas (skat. 6.13. att.). Lai atkārtoti iedarbinātu taimeri,

6.13. att. **Taimera palaišana**

ir jāizslēdz I0.1 „0" un jāieslēdz atkārtoti „1", vai uz I0.1 jādod palaišanas impulsu „1". Laikā, kamēr taimeris ir ieslēgts, kontakti T1 ir saslēgti un uz izejas O0.1 ir „1" – lampiņa deg (skat. 6.14. att.).

6.14. att. **Taimera izeja**

Ja nepieciešams taimeri izslēgt jebkurā laikā, tad jāizveido zars ar RESET operāciju T1 (R). Padodot uz ieeju I0.2 „1", nostrādā taimera T1 RESET operācija (R) un taimeri T1 izslēdz (skat. 6.15.att.).

6.15. att. **Taimera izslēgšana**

Ja nepieciešams taimeri ieslēgt, to var izdarīt izmantot arī SET operāciju T1 (S), ar speciāli izveidotu programmas zaru (skat. 6.16. att.). Ja padod „1" uz I0.3 ieeju, nostrādā operācija SET (S) un taimeris sāk skaitīt laiku, līdz sasniedz iestatīto vērtību

6.16. att. **Taimera ieslēgšana**

5 sekundes un izslēdzas. Ja ir sasniegts izslēgšanās laiks, taimeris izslēdzas neatkarīgi no tā, vai uz I0.3 ir „0", vai ir saglabājies neizslēgts „1".

6.17. att. **Taimera vadība**

Ja virknē ar taimera barošanu T1 ir ieslēgts viņa paša normāli slēgtais kontakts T1, tad taimeris dos īsu izslēgšanās impulsu ik pēc 5 sekundēm, bet pats paliks ieslēgtā stāvoklī tik ilgi, kamēr uz ieejas I0.1 būs „1" un vēl 5 sekundes, ja uz šīs ieejas pados īsu palaišanas impulsu, tad pēc 5 sekundēm taimeris izslēgsies.

Skaitītājs

Skaitītājs var skaitīt impulsus, ieslēgt un izslēgt ķēdi. Skaitītājs skaita uz INC vai DEC elementu padotos impulsus.

Katram skaitītājam ir adrese **C1; C2; C3 ... Cnn**.

CPnn (counter preset) iestata no kāda skaitļa skaitīs ;

CWnn (counter word) vārds, skaitītāja momentānais lielums*;

Cnn skaitītāja loģiskais statuss 0 vai 1.

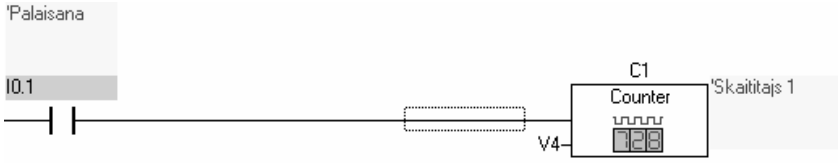
C=1 notiek skaitīšana - skaitītājs ieslēgts;

C=0 skaitītājs beidzis skaitīt, vai nav sācis skaitīt – skaitītājs izslēgts.

**Procesu vadībai var izmantot skaitītāja momentānās vērtības CWnn. Skaitītājs, pēc momentānās vērtības, skaita no iestatītās sākuma vērtības līdz nullei un tad izslēdzas! Faktiski INK nevis pieskaita, bet atņem.*

Skaitītāja programmēšana

Ja uz I0.1 ieeju padodam „1", vai iedodam pozitīvu „1" impulsu, sāk darboties skaitītājs C1. Skaitītājam ir iestatīta vērtība V4, tas nozīmē, ka skaitītājs skaitīs līdz 4 un tad izslēgsies. Lai atkārtoti ieslēgtu skaitītāju, ir jāizslēdz I0.1 „0" un jāieslēdz atkārtoti „1", vai uz I0.1 jādod palaišanas impulsu „1" (skat. 6.18. att.).



6.18. att. **Skaitītāja palaišana**

Laikā, kamēr skaitītājs ir ieslēgts un skaita, kontakti C1 ir saslēgti un uz izejas O0.1 ir „1" – lampiņa deg (skat. 6.19. att.).

6.19. att. **Skaitītāja izeja**

Skaitītājs pieskaita impulsus, kurus padodam uz speciāli izveidotu skaitītāja C1 increment INC ķēdi (I) (skat. 6.20. att.). Ja uz ieeju I0.4 padodam „1", tad skaitītājs piesummē vienu vienību. Ja 4 reizes padodam „1" uz I0.4, skaitītāja saskaitītā impulsu

summa ir 4 un tā ir vienāda ar iestatīto vērtību V4 – skaitītājs izslēdzas, izejā O0.1 pieslēgtā lampiņa nodziest.

6.20. att. **Pieskaitīšana**

Ar operāciju decrement DEC notiek atņemšana. Izveidojam skaitītāja C1 dekrementa ķēdi (D) , padodot „1" uz ieeju I0.5, no skaitītāja summas (mainīgā lieluma) atņem vienu vienību (skat. 6.21. att.). Ja padodam uz ieeju I0.5 trīs impulsus, tad skaitītāja summa būs –3. Šajā gadījumā, lai skaitītājs izslēgtos uz incrementa elementu (I) ir jāpadod 7 impulsi, tad summa būs 4 = V4, (-3+7=4).

6.21. att. **Atņemšana**

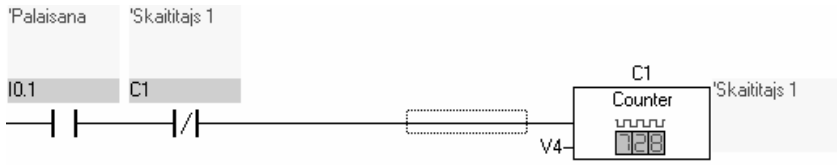
Jebkurā momentā skaitītāju var izslēgt, ja ir izveidots elements RESET **(**R**)**. Ja uz ieejas I0.2 padodam „1", tad skaitītājs C1 momentāni tiek izslēgts (skat. 6.22. att.).

6.22. att. **Skaitītāja izslēgšana**

Skaitītāja ieslēgšanai var lietot ieslēgšanas elementu SET (S). Ja uz ieejas I0.3 padodam „1", tad skaitītājs C1 tiek ieslēgts (skat. 6.23. att.).

6.23. att. **Skaitītāja ieslēgšana**

Ja skaitītājs jau ir ieslēgts, tad ar I0.3 izpildot operāciju SET (S), skaitītājs atkārtoti tiek ieslēgts un skaitīšana sākas no sākuma.



6.24. att. **Skaitītāja palaišana**

Ja skaitītāja ķēdē ieslēdz viņa paša normāli slēgtos kontaktus, tad tas paliek ieslēgts tik ilgi, kamēr uz ieejas I0.1 ir „1" un skaitītāja summa ir 4+4+…+4. Skaitītājs saskaita līdz 4, tad normāli slēgtais kontakts palaiž skaitītāju atkārtoti un skaitītājs paliek ieslēgts, tas turpinās, kamēr atslēdz ieeju I0.1 un pēdējā summa sasniedz 4 (skat. 6. 24. att.).

6.3. Programmēšana ar STL valodu

Šai nodaļā, kā piemērs, aprakstīta STL programmēšanas valoda, kas ir izmantota FST programmatūrā un paredzēta FESTO kontrolleru programmēšanai. STL ir teksta valoda programmējamo kontrolleru programmēšanai. Šo valodu labprāt izvēlās izmantot neelektriķi, jo nav nepieciešamas speciālās zināšanas elektrisko shēmu projektēšanā.

Lai varētu lietot STL valodu, nepieciešams zināt programmas pieraksta sintaksi, komandas un operācijas (skat. 6.1. tabulā).

Programma tiek veidota pēc principa, ja (IF) ir izpildījies nosacījums, tad (THEN) kaut kas notiek. Ja (IF) ieejā (input I0.0) ir loģiskais vieninieks, tad (THEN) ieslēdz (SET) izeju (output O0.1). Kad ir izpildīta pirmā komanda, ***tikai tad*** izpilda nākošo.

6.1. tabula

STL valodas komandu un operāciju saraksts

Operācija	**Apraksts**
STEP	Norāda programmas secību, sadala pa soļiem Piemēram: **STEP 1** **……** **THEN JMP TO 1**
IF	Salīdzināšanas funkcija „ja” Piemēram: **IF I0.1** **AND N I0.2** **…….**
THEN	Sekojoša operācija, ja IF nosacījumi piepildās (atbilst) „tad” Piemēram: **THEN LOAD V100** **TO TP7** **……..**
OTHRW	Sekojoša operācija, ja nepiepildās IF nosacījumi, „tādā gadījumā” Piemēram: **THEN SET O0.0** **OTHRW RESET O0.2**

NOP	Neveikta operācija. Šis ieejas nosacījums vienmēr pareizs Piemēram: **IF NOP** **THEN SET F0.1**
JMP TO	Pāreja uz norādīto programmas vietu, parasti seko aiz THEN vai OTHRW Piemēram: **STEP 1** **……** **STEP12** **IF I0.2** **THEN SET O0.3** **JMP TO 1**
SET	Viena bita izejas operandus ieslēdz loģiskā „1"stāvoklī Startē (ieslēdz) programmas, taimerus, skaitītājus.
RESET	Viena bita izejas operandus ieslēdz loģiskā „0"stāvoklī Apstādina (izslēdz) programmas, taimerus, skaitītājus.
LOAD	Ielādē konstantes, taimeriem, skaitītājiem, reģistriem, uzkrāj bitus Piemēram: **THEN LOAD V500** **TO TP31**
SWAP	Darbība ar vārdu, vārda pārvietošana mainot 8…15 bitus uz 0…7, Piemēram: **THEN LOAD VS55AA** **TO OW0** **SWAP** **TO OW1**
ROL	Rotācija pa kreisi (Rotate to Left). Bits 15 (MSB) nonāk 0 (LSB) bita vietā. Notiek vārda pārbīdīšana pa vienu šūnu pa

	kreisi.
ROR	Rotācija pa labi (Rotate to Right). Bits 0 (LSB) nonāk 15 (MSB) bita vietā. Notiek vārda pārbīdīšana pa vienu šūnu pa labi.
SHL	Pārbīde pa kreisi (Shift to Left). Bits 0 (LSB) tiek nonullēts, pārslēgts uz loģisko 0. Notiek vārda pārbīdīšana pa vienu šūnu pa kreisi. Piemēram: **THEN LOAD V16** **SHL** **TO R7**
SHR	Pārbīde pa labi (Shift to Right). Bits 15 (MSB) tiek nonullēts, pārslēgts uz loģisko 0. Notiek vārda pārbīdīšana pa vienu šūnu pa labi. Piemēram: **THEN LOAD V16** **SHR** **TO R7**
BID	Bināro vārdu konvertē uz binări decimālo (BCD) Piemēram: **THEN LOAD IW0** **BID** **TO OW7**
DEB	Binări decimālo vārdu (BCD) konvertē uz bināro Piemēram: **THEN LOAD IW7** **DEB** **TO OW7**
(……..)	Operandu grupas ietveršana iekavās
+	Summē konstantus lielumus

-	Atņem konstantus lielumus
*	Reizināšana
/	Dalīšana
<	Mazāks
<=	Mazāks un vienāds
=	Vienāds
>=	Lielāks un vienāds
>	Lielāks
<>	Nevienāds
AND	Loģiskā funkcija UN Piemēram: **IF I0.2** **AND I0.3** **THEN SET O0.1** **OTHRW RESET O0.4**
OR	Loģiskā funkcija VAI Piemēram: **IF I0.1** **OR I0.2** **OR I0.4** **THEN SET O0.2** **OTHRW RESET O0.7**
EXOR	Loģiskā funkcija Izslēdzošais VAI (EXCLUSIVE OR) Piemēram: **IF I0.1** **EXOR I0.2** **THEN SET O0.2** **OTHRW RESET O0.7**
TO	Lieto kopā ar LOAD, norāda kur ielādēt Piemēram:

	THEN LOAD V500 **TO R5**
INC	Increment word (+1). Pieskaita vienīnieku.
DEC	Decrement word (-1). Atņem vienīnieku.
N	Noliegums, ja nav loģiskais "1". Ja ir "0". Piemēram: **IF N O0.2** **THEN SET F0.3**

Programmas rakstīšana

Programmu var rakstīt nepārtrauktu vai pa soļiem STEP 1, STEP x, STEP 4 utt. Kad pirmā soļa programma ir izpildīta, neatkatrīgi no soļa apzīmējuma notiek pāreja uz nākošo soli. Ja norādīs pāreju uz citu soli, piemēram, JMP TO 4, tad notiks pāreja uz STEP 4.

Rakstot STL programmu to var papildināt ar komentāriem lietojot "", piemēram.

```
STEP 1
    ""komentārs… Ieslēdzam lampiņu
            IF      I0.0    'start
    THEN            O0.0    'lampina
STEP 2
    ""komentārs…Izsledzam lampinu
            IF      I0.1    'stop
    THEN    RESET   O0.0    'lampina
STEP 3
    ""komentārs…Atgriesties uz programmas sākumu
            IF      NOP
    THEN    JMP TO 1
```

Izejas vārda iestatīšana

Lai visās kontrollera izejās O0.0 līdz O0.7 būtu 0, var iestatīt izeju reģistra vārdu OW0 ar 0 vērtību, kas atbilst binārajam skaitlim 00000000.

```
IF      ……
THEN    LOAD    V0
TO              OW0
```

Rezultātā visās izejās būs nulle.

Ja nepieciešams visās kontrollera izejās iestatīt vienīnieku, ieslēgt izejas O0.0 līdz O0.7, tad izejas reģistra vārdu OW0 iestata ar vērtību 255, kas atbilst binārajam skaitlim 11111111. Parasti izmanto 8 bitu vārdu. Tas atbilst astoņām izejām.

```
IF      …….
THEN    LOAD    V255
TO              OW0
```

Rezultātā visas izejas būs ieslēgtas.

Izejas reģistrā OW0 var ierakstīt jebkuru decimālo skaitli no 0 līdz 255, atkarībā no šī skaitļa, mainīsies tikai ieslēgto izeju kombinācija, kas atbilst binārajam skaitlim.

Binārais un decimālais skaitlis

Mēs esam pieraduši izmantot decimālo skaitīšanas sistēmu, tāpēc vēlamies ieraudzīt šīs sistēmas skaitļus. Ciparu tehnikā datu apstrāde notiek binārā formā, ieslēgts, izslēgts, nulle, vienīnieks. Lai varētu izmantot un novērtēt šos bināros datus dažreiz ir nepieciešams tos pārvērst decimālā formā.

Cipars ir rakstu zīme, kas attēlo skaitli, skaitlis sastāv no cipariem. Tāpat, kā decimālajā sistēmā, arī binārajā sistēmā, cipara atrašanās vietai ir sava nozīme un svars. Decimālajā sistēmā šo svaru nosaka skaitļa 10 pakāpes – kārta, piemēram:

$10^0 = 1$ nulltā kārta – vienīnieki;

$10^1 = 10$ pirmā kārta – desmiti;

$10^2 = 100$ otrā kārta – simti utt.

Bināro skaitļu sistēmā, katra kārta tiek noteikta, ņemot par bāzi skaitļa 2 pakāpi:

$2^0 = 1$

$2^1 = 2$

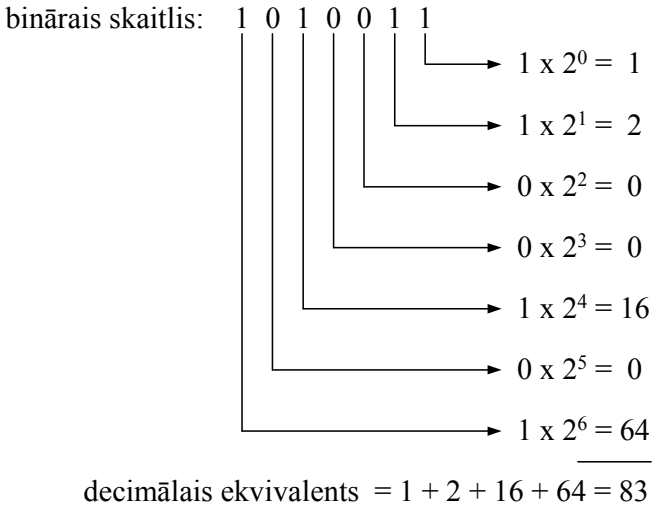
$2^2 = 4$ utt.

Binārā skaitļa pamatvienība ir bits. Binārais skaitlis var sastāvēt no atsevišķiem bitiem, vai bitu virknes, no bināriem cipariem veidotas n kārtas skaitļa, kur katra kārta var pieņemt vērtību 1 vai 0.

Binārā skaitļa pēdējais labās puses bits ir jaunākās kārtas bits (LSB – least significant bit). Ja binārais skaitlis ir vesels skaitlis, tad jaunākās kārtas bits LSB ir $2^0 = 1$ un katrs bits pa kreisi ir par vienu kārtu lielāku vērtību.

Binārā skaitļa pēdējais kreisās puses bits ir vecākās kārtas bits (MSB – most significant bit).

Lai pārvērstu bināru skaitli ekvivalentā decimālajā, ir jāaprēķina visu binārā skaitļa kārtu decimālā vērtība un jāsasummē (skat. 6.25. att.).



6.25. att. **Binārā skaitļa pārveidošana decimālajā**

Vienkāršākais veids, kā konvertēt bināro skaitli uz decimālo ekvivalentu, ir uzrakstot binārā skaitļa kārtu decimālās vērtības ...16, 8, 4, 2, 1 virs binārā skaitļa katra

atbilstošas kārtas bita, sākot ar jaunāko kārtu LSB un beidzot ar vecāko kārtu MSB, un summējot to bitu decimālās vērtības, kuru statuss ir 1 (skat. 6.26. att.).

2 pakāpe: 16 8 4 2 1

binārais skaitlis: 1 0 1 1 0

decimālais skaitlis = 16 + 4 + 2 = 22

6.26. att. **Binārā skaitļa 10110 pārvēršana decimālajā**

Matemātisko darbību pielietošana

Programmās var būt izmantotas matemātiskās darbības.

Piemēram:

```
IF      (       FW0
        =       V123 )
        AND
        (       R1
        <>      V0
        )
        THEN ……..
```

Taimera programmēšana

Taimeri raksturo vairāki parametri.

1. Taimera statuss (Tnn, kur nn ir no 0 līdz 255, atkarībā no kontrollera tipa) ja Tnn=0, tad taimers ir neaktīvs, ja iestatītais laiks ir beidzies, tad Tnn=1. Taimera statuss ir iestatāms.
2. Taimera iestatīšana (TPnn, kur nn ir no 0 līdz 255)(Timer Preset). Iestatāmais laika periods ir no 0.00 s līdz 655.35 s.
3. Taimera vārds (TWnn, kur nn ir no 0 līdz 255).

Piemēram:

Taimera inicializēšana. Uzliekam laiku 5.2 s

```
IF      …….
THEN    LOAD    V520
        TO      TP7
```

Palaižam taimeri

```
IF      ……
THEN    SET     T7
```

Vai arī,

```
IF      ……
THEN    SET     T9
        WITH    3s
```

Taimera apstādināšana

```
IF              I0.1
        AND     I1.1
        AND     T7
THEN    RESET   T7
OTHRW   ……
```

Taimera izmantošana slēgšanai

```
IF      N       T7
THEN    SET     O0.2
```

Skaitītāja programmēšana

Skaitītājs var strādāt pieskaitīšanas vai atņemšanas režīmā. Skaitītāju raksturo vairāki parametri.

1. Skaitītāja statuss (Cnn, kur nn var būt no 0 līdz 255, atkarībā no kontrollera tipa)
   Ja Cnn = 0, tad skaitītājs ir neaktīvs.

Ja Cnn=1, tad skaitītājs ir aktīvs, tas skaita.

Skaitītāja statusu var iestatīt.

2. Skaitītāju iestata (CPnn, kur nn no 0 līdz 255)

   Skaitlis, ko skaita var būt no 1 līdz 65535.

3. Skaitītāja vārds (CWnn, kur nn no 0 līdz 255)

Piemēram:

Inkrementa (pieskaitīšanas) skaitītāja inicializēšana.

```
IF      …….
THEN        LOAD        V100
            TO      CP15
            SET     C15
```

Dekrementa (atskaitīšanas) skaitītāja inicializācija.

```
IF      …….
THEN        LOAD        V100
            TO      CP15
…….
```

Pieskaitīšana (inkrements)

```
…….
THEN        INC     CW0
THEN        INC     C0
…….
```

Atņemšana (dekrements)

```
…….
THEN        DEC     CW0
THEN        DEC     C0
…….
```

Skaitītāja apturēšana

```
IF      …….
THEN    RESET   C15
…….
```

Skaitītāja izmantošana slēgšanai.

```
IF      N       C15
THEN    SET     O0.3
…….
IF      C3
THEN    SET     O0.2
OTHRW   RESET   O0.2
…….
IF      (       CW15
        =       V25
        )
THEN    LOAD    V170
        TO      TP7
        SET     T7
……
```

7. PROGRAMMĒŠANAS PIEMĒRI

7.1. Kāpņu diagrammu programmas

1. Piemērs. Vadība ar Start un Stop pogām

Izveidosim programmu, kur nospiežot pirmo pogu lampiņa tiek ieslēgta, bet nospiežot otro pogu izslēgta.

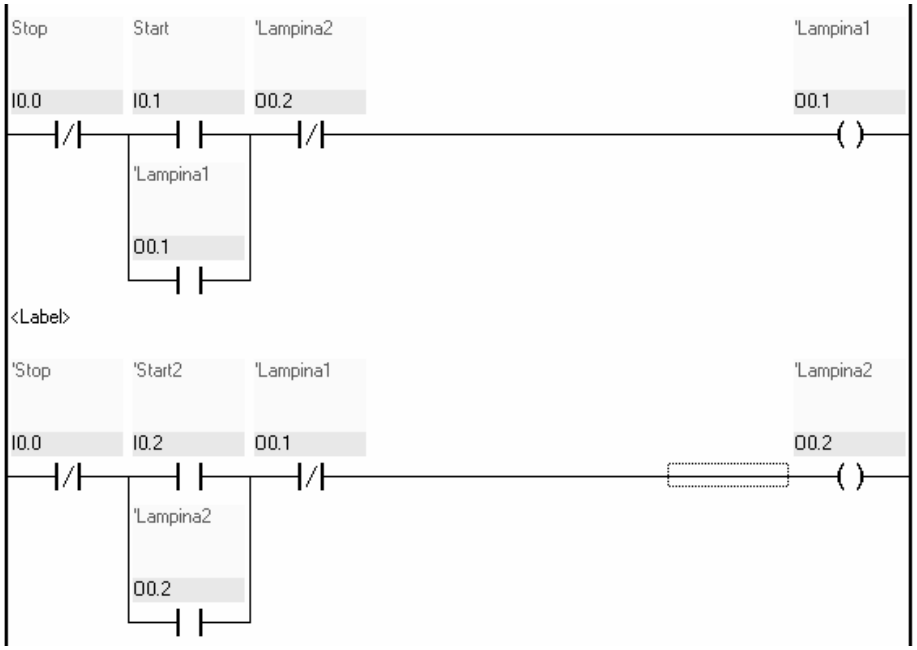
7.1. att. **Vadība ar pogām Start un Stop**

Programma ir veidota no četriem elementiem, trīs kontaktiem un spoles (skat. 7.1. att.). „Sart" pogas kontakti ir definēti kā kontrollera ieeja I0.1, „Stop" pogas kontakti ir definēti kā ieeja I0.0, bet izejas ķēde ar apzīmējumu „Lampiņa 1" ir spole (coil), tā definēta kā kontrollera izeja O0.1.

Iepazīsimies ar programmas darbību. Sākotnēji ķēde I0.0, I0.1, O0.1 ir pārtraukta, jo kontakti I0.1 ir atvērti. Izejas O0.1 statuss ir 0, lampiņa nedeg. Nospiežot pogu „Start" kontakti I0.1 saslēdzas un ķēde noslēdzas, tiek ieslēgta izeja O0.1, iedegas lampiņa. Atlaižot pogu „Start", izeja O0.1 paliek ieslēgta, to nodrošina noturkontakti O0.1, kas ir pieslēgti paralēli „Start" pogas kontaktiem I0.1. Paralēlais zars ar kontaktiem O0.1 kalpo kā atmiņas elements. Nospiežot pogu „Stop" kontakti I0.0 atveras, ķēde tiek pārtraukta un izeja O0.1 izslēdzas, lampiņa nodziest.

2. Piemērs. Savstarpējā bloķēšana

Izveidosim programmu ar divu izeju savstarpējo bloķēšanu. Ja ieslēdz vienu no izejām, tad otru ieslēgt nav iespējams.



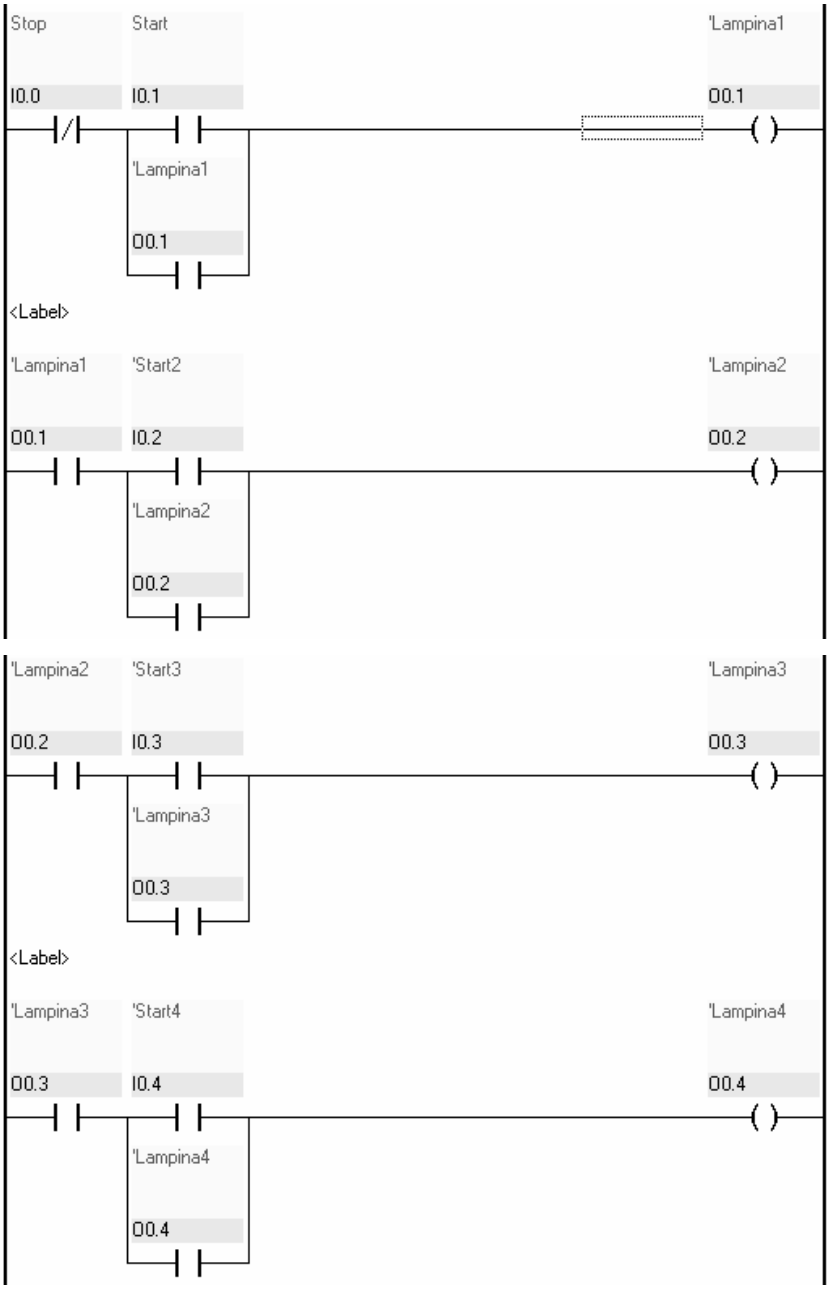
7.2. att. **Savstarpējā bloķēšana**

Programma sastāv no diviem vienādiem zariem. Lai nodrošinātu savstarpējo bloķēšanu, katrā zarā ir ieslēgti normāli slēgtie bloķējošie kontakti, kurus darbina otra zara izeja (skat. 7.2. att.).

Programma darbojas šādi. Nospiežot pogu „Start", kontakti I0.1 saslēdzas, tiek ieslēgta pirmā izeja O0.1, rezultātā, otrajā zarā normāli slēgtie kontakti O0.1 atveras un otro izeju O0.2 ieslēgt vairs nav iespējams. Lai ieslēgtu otro izeju O0.2, ir jāizslēdz pirmā. Nospiežam pogu „Stop", kas ar kontaktiem I0.0 pārtrauc pirmā zara ķēdi un izslēdz pirmo izeju O0.1. Tikai tad, ja ir izslēgta pirmā izeja O0.1, var ieslēgt otro izeju O0.2. Nospiežam pogu „Start2", kontakti I0.2 saslēdzas, tiek ieslēgta otrā izeja O0.2, rezultātā, pirmajā zarā atveras normāli slēgtie kontakti O0.2 un pirmo izeju O0.1 ieslēgt vairs nav iespējams.

3. Piemērs. Secīgā bloķēšana

Izveidosim programmu, kas atļauj ieslēgt ieejas tikai noteiktā secībā: 1, 2, 3, 4, bet izslēgšana notiek vienlaicīgi, ar vienas pogas palīdzību.



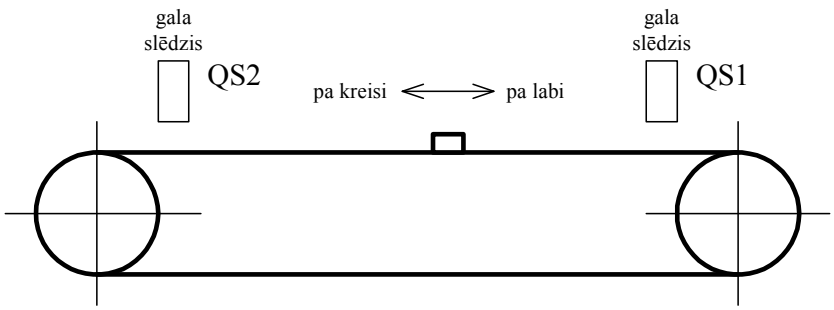
7.3. att. **Secīgās bloķēšanas programma**

Programma sastāv no četriem vienādiem zariem. Lai nodrošinātu secīgo bloķēšanu, katram nākošajam zaram virknē ir ieslēgti normāli atvērtie kontakti, kurus darbina iepriekšējā zara izeja (skat. 7.3. att.).

Iepazīsimies ar programmas darbību. Sākotnēji ķēde I0.0, I0.1, O0.1 ir pārtraukta, jo kontakti I0.1 ir atvērti, izeja O0.1 ir izslēgta. Nospiežot pogu „Start" kontakti I0.1 saslēdzas un ķēde noslēdzas, tiek ieslēgta izeja O0.1. Atlaižot pogu „Start", izeja O0.1 paliek ieslēgta, to nodrošina noturkontakti O0.1, kas ir pieslēgti paralēli „Start" pogas kontaktiem I0.1. Pirmā izeja O0.1 ir ieslēgta, slēgt nākošo zaru ļauj virknē ieslēgtie kontakti O0.1, kas arī ir saslēgti. Nospiežot pogu „Start2" kontakti I0.2 saslēdzas un ķēde noslēdzas, tiek ieslēgta izeja O0.2, slēgt nākošo zaru ļauj virknē ieslēgtie kontakti O0.2, kas arī ir saslēgti. Nospiežot pogu „Start3" kontakti I0.3 saslēdzas un ķēde noslēdzas, tiek ieslēgta izeja O0.3, slēgt nākošo zaru ļauj virknē ieslēgtie kontakti O0.3, kas arī ir saslēgti. Nospiežot pogu „Start4" kontakti I0.4 saslēdzas un ķēde noslēdzas, tiek ieslēgta izeja O0.4. Nospiežot pogu „Stop" kontakti I0.0 atveras, ķēde tiek pārtraukta un izejas O0.1, O0.2, O0.3, O0.4 izslēdzas.

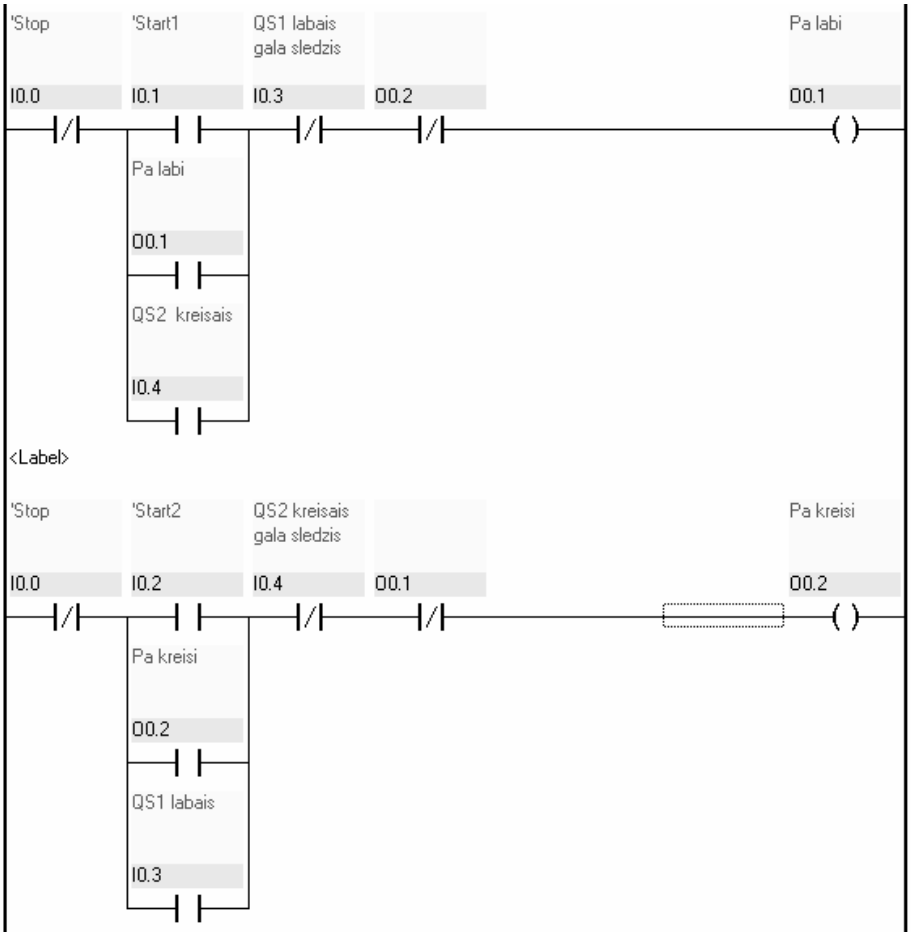
4. Piemērs. Reversīvais transportieris

Izveidosim reversīvā transportiera vadības programmu, kur uz lentas novietotais priekšmets pārvietosies cikliskā lineārā kustībā pa labi un pa kreisi. Transportiera kustības virzieni un galaslēdžu izvietojums dots 7.4. attēlā.



7.4. att. **Reversīvais transportieris**

Programma sastāv no diviem vienādiem zariem, pirmais zars nodrošina transportiera kustību pa labi, otrais kustību pa kreisi. Lai nodrošinātu savstarpējo bloķēšanu, katrā zarā ir ieslēgti normāli slēgtie bloķējošie kontakti, kurus darbina blakus zara izeja (skat. 7.5.att.).



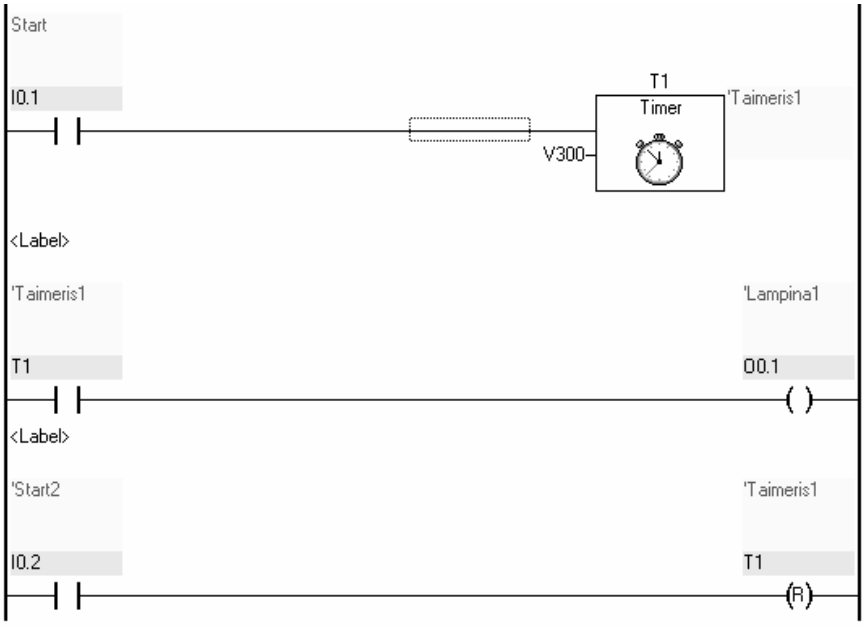
7.5. att. **Reversīvā transportiera vadības programma**

Programma darbojas šādi. Nospiežot pogu „Start", kontakti I0.1 saslēdzas, tiek ieslēgta pirmā izeja O0.1, transportieris pārvietojas pa labi, līdz sasniedz labās puses galaslēdzi QS1. Tas nostrādā un atver normāli slēgtos kontaktus I0.3, rezultātā izeju

O0.1 izslēdz, transportieris apstājas. Nostrādājot, galaslēdzis QS1 saslēdz otrā zara kontaktus I0.3, tiek ieslēgta otrā izeja O0.2, transportieris pārvietojas pa kreisi, līdz sasniedz kreisās puses galaslēdzi QS2. Tas nostrādā un atver normāli slēgtos kontaktus I0.4, rezultātā izeju O0.2 izslēdz, transportieris apstājas. Nostrādājot, galaslēdzis QS2 saslēdz pirmā zara kontaktus I0.4, tiek ieslēgta pirmā izeja O0.1, transportieris pārvietojas pa labi, līdz sasniedz labās puses galaslēdzi QS1. Šādā veidā cikli turpinās bezgalīgi. Lai apstādinātu transportieri, jānospiež pogu „Stop" kontakti I0.0 atveras, pirmā vai otrā zara ķēde tiek pārtraukta un transportieris apstājas.

5. Piemērs. Laika relejs

Izveidosim programmu, kur nospiežot pogu lampiņa iedegas un deg 3 sekundes, bet ja nepieciešams, lampiņu var izslēgt arī ātrāk.



7.6. att. **Laika relejs**

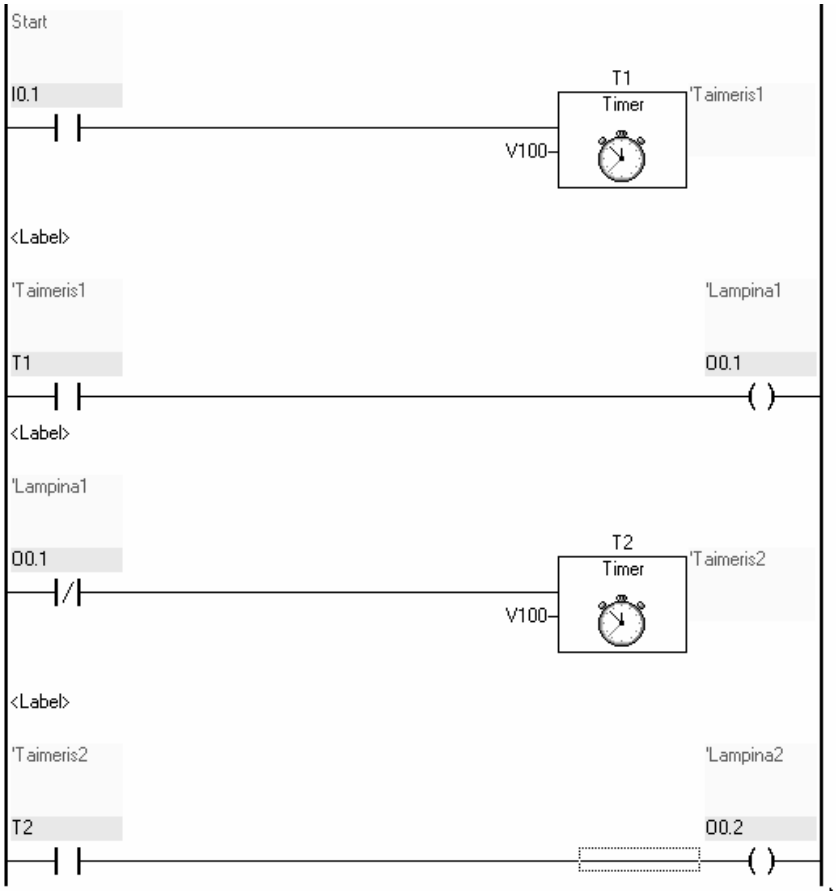
Programmā ir zars ar taimeri T1, kuram laika kavējums ir 3 sekundes, tā palaišanai paredzēti kontakti I0.1. Lai darbinātu lampiņu izmantots atsevišķs zars ar

izeju O0.1, kuru slēdz taimeris. Taimera priekšlaicīgai izslēgšanai izveidots zars ar pogu un RESET elementu (R) (skat. 7.6. att.).

Programma darbojas šādi. Nospiežot pogu „Start", kontakti I0.1 saslēdzas, tiek palaists taimeris T1. Kamēr taimeris darbojas, kontakti T1 ir saslēgti, izeja O0.1 ir ieslēgta, lampiņa deg. Kad taimera iestatītais laiks 3 sekundes ir beidzies, taimeris izslēdzas, lampiņa nodziest. Saslēdzot kontaktus I0.2, taimeri T1 var izslēgt jebkurā momentā. Lai iedegtu lampiņu no jauna, nepieciešams nospiest pogu „Start".

6. Piemērs. Pārslēdzējs ar diviem taimeriem

Izveidosim programmu, kur nospiežot pogu lampiņa iedegas un deg 1 sekundi, tad nodziest un iedegas otra lampiņa un deg 1 sekundi (skat. 7.7. att.).



7.7. att. **Pārslēdzējs ar diviem taimeriem**

Programmā izmantoti divi taimeri pirmais taimeris T1 slēdz pirmo lampiņu, otrais taimeris T2 slēdz otro lampiņu. Pirmais un otrais taimeris ir saslēgti secīgā slēgumā, kad beidzas pirmā taimera laika aizture, tas iedarbina otru taimeri. Ar šādu struktūru var izveidot programmu arī ar lielāku slēdzamo izeju skaitu.

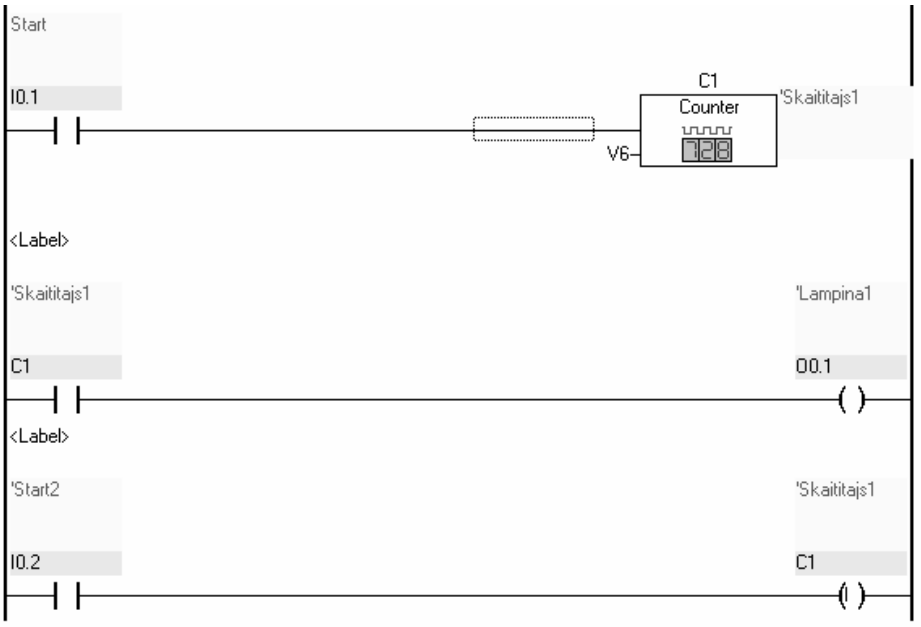
Iepazīsimies ar programmas darbību. Nospiežot pogu „Start" kontakti I0.1 saslēdzas, tiek ieslēgts pirmais taimeris T1. Kamēr taimeris T1 ir ieslēgts, T1 kontakti ir saslēgti un izeja O0.1 ieslēgta, pirmā lampiņa deg. Kad ir pagājusi viena sekunde, lampiņa tiek izslēgta un uzreiz tiek ieslēgts otrais taimeris. Uz vienu sekundi iedegas otrā lampiņa, kas ir pievienota kontrollera O0.2 izejai. Otrā taimera ieslēgšanu izpilda normāli saslēgtie kontakti O0.1, kas pēc iepriekšējā cikla, kad bija atvērti, saslēdzas. Lai atkārtotu lampiņu mirgošanas ciklu, ir jānospiež poga „Start", kas saslēdz kontaktus I0.1.

7. Piemērs. Impulsu skaitītājs

Izveidosim programmu, kur nospiežot vienu pogu lampiņa iedegas, nospiežot otru pogu sešas reizes, lampiņa nodziest.

Programmā ir zars ar skaitītāju C1, kas skaita līdz 6, tā palaišanai paredzēti kontakti I0.1. Lampiņas ieslēgšanai izmantots atsevišķs zars ar izeju O0.1, to slēdz skaitītājs C1. Skaitītājs saskaita kontaktu I0.2 saslēgšanās reižu skaitu (skat. 7.8. att.).

Iepazīsimies ar programmas darbību. Nospiežot pogu „Start", saslēdzas kontakti I0.1 un tiek iedarbināts skaitītājs C1. Kamēr skaitītājs ieslēgts, kontakti C1 ir saslēgti un izeja O0.1 ieslēgta, lampiņa deg. Katru reizi, kad tiek nospiesta otrā poga „Start2", saslēdzas kontakti I0.2 un skaitītāja vērtība tiek palielināta par vienu vienību. Skaitītājs paliek ieslēgtā stāvoklī, kamēr saskaita līdz 6, sasniedzot šo vērtību, tas izslēdzas un izslēdz arī lampiņu.

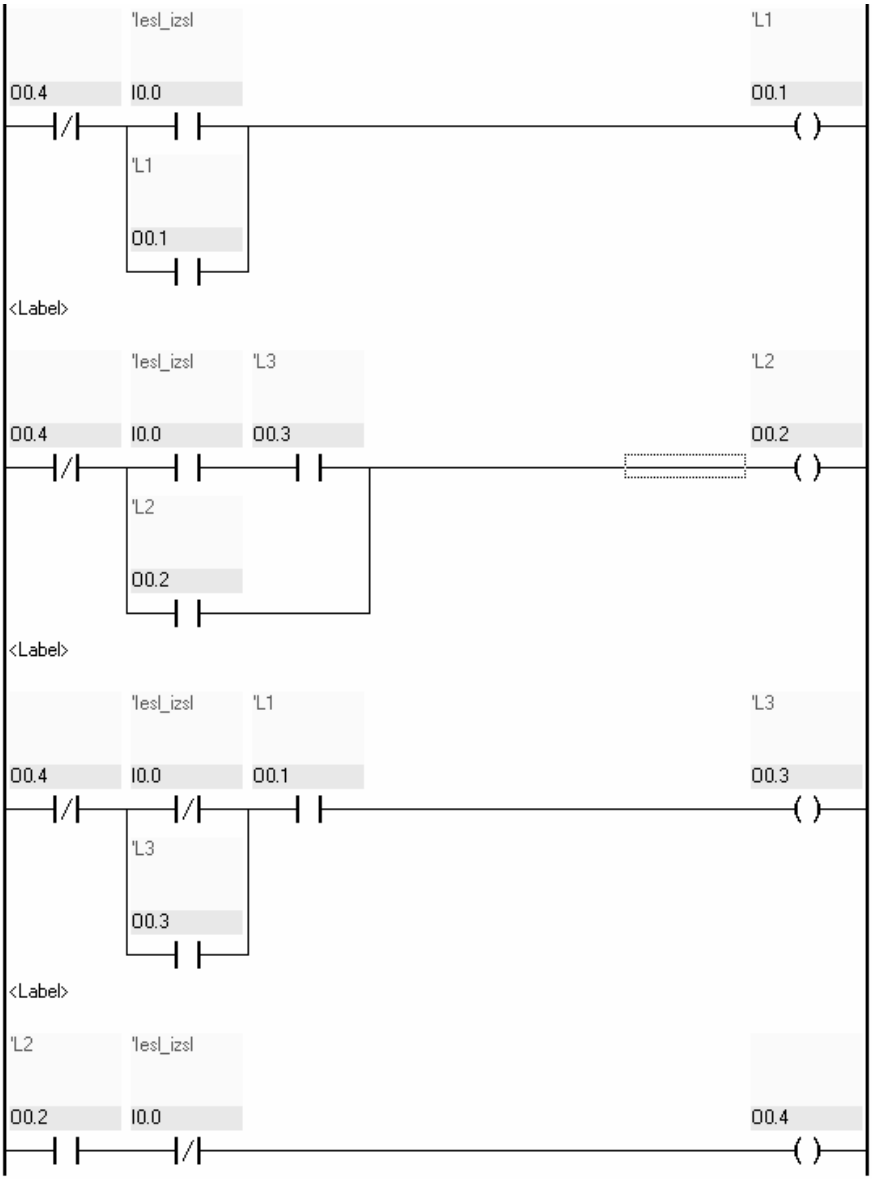


7.8. att. **Impulsu skaitītājs**

8. Piemērs. Četru taktu slēdzis

Izveidosim programmu, kas izpilda ieslēgšanu un izslēgšanu ar vienu un to pašu pogu.

Programma darbojas šādi. Nospiežot un neatlaižot pogu tiek iedegta pirmā lampiņa „L1", tā iedegas uz pogas nospiešanu. Atlaižot pogu iedegas trešā lampiņa „L3", to ieslēdz pogas normāli slēgtie kontakti trešajā zarā, kur nospiežot pogu tika saslēgti kontakti O0.1 un atlaižot pogu saslēgts arī I0.0, kā rezultātā tiek ieslēgta izeja O0.3. Vēlreiz nospiežot pogu otrajā zarā tiek ieslēgts O0.2, jo O0.3 jau ir ieslēgts. Atlaižot pogu tiek ieslēgts O0.4, kas atbild par visas shēmas izslēgšanu, kā rezultātā tiek nodzēstas visas lampiņas. Veidojot programmu izeju O0.2, O0.3, O0.4 vietā vēlams izmantot viena bita atmiņas F0.2, F0.3, F0.4, mūsu vajadzībām, to skaits nav ierobežots (skat. 7.9. att.).



7.9. att. **Četru taktu slēdzis**

7.2. Programmas STL valodā

9. Piemērs

Izveidosim programmu, kur atkarībā no nospiestās pogas notiek lampiņu iedegšana uz 1 sekundi.

```
IF                    NOP
THEN
      LOAD            V100
        TO            TP1
IF                    I0.0
THEN  SET             O0.0            'nulta lampina
      SET             T1              'taimeris
IF            N       T1              'taimeris
THEN
      RESET           O0.0            'nulta lampina
IF                    I0.1
THEN  SET             O0.1            'Lampina
      SET             T1              'taimeris
IF            N       T1              'taimeris
THEN  RESET           O0.1            'Lampina
IF                    I0.2            'poga2
THEN  SET             O0.2
      SET             T1              'taimeris
IF            N       T1              'taimeris
THEN  RESET           O0.2
IF                    I0.0
      AND             I0.1
THEN
      SET             O0.3
      SET             T1              'taimeris
      RESET           O0.0            'nulta lampina
      RESET           O0.1            'Lampina

IF            N       T1              'taimeris
THEN  RESET           O0.3
```

Programma rakstīta vienlaidu kodā, neizmantojot soļus. Ar trīs pogām var iedegt atbilstošo lampiņu uz 1 sekundi, nulltā poga I0.0 ieslēdz nullto lampiņu O0.0, pirmā poga I0.1 pirmo lampiņu O0.1, otrā poga I0.2 otro lampiņu O0.2. Ja tiek nospiesta nulltā un pirmā poga reizē , tad tiek iedegta 3. lampiņa.

Programmas sākumā tiek definēts taimeris T1 ar laika iestatījumu V100 = 1 sekunde. Ja programmas izpildes laikā piepildās kāds no nosacījumiem IF, piemēram, ir nospiesta nultā poga un saslēgti kontakti I0.0, tiek ieslēgta izeja O0.0 un taimeris T1 (THEN SET O0.0 SET T1). Kad taimeris ir beidzis darboties (IF N T1), izslēdz izeju O0.0 (THEN RESET O0.0).

10. Piemērs

Izveidosim programmu, kas atkarībā no nospiestās pogas izpilda lampiņu secīgu iedegšanos un nodzišanu.

Ja tiek nospiesta „Start1” poga, tad izpilda programmas daļu no soļa 1a līdz solim 2a, viena pēc otras secīgi nomirgo lampiņas 1-2-3. Ja tiek nospiesta „Start2”

poga, tad tiek izpildīti soļi no 2a līdz programmas beigām, viena pēc otras secīgi nomirgo lampiņas 3-2-1. Katra cikla beigās programma atgriežas pie pirmā soļa, kur gaida, kura poga tiks nospiesta.

```
STEP 1
 IF                    NOP
 THEN  LOAD            V30
         TO            TP1
STEP 2
 IF                    I0.0            'Start1
 THEN  JMP TO 1a
 IF                    I0.1            'Start2
 THEN  JMP TO 2a
STEP 1a
 IF                    I0.0
 THEN  SET             O0.1            'Lampina
       SET             T1              'taimeris
STEP 4
 IF             N      T1              'taimeris
 THEN  SET             O0.2
       SET             T1              'taimeris
STEP 5
 IF             N      T1              'taimeris
 THEN  SET             O0.3
       SET             T1              'taimeris
STEP 6
 IF             N      T1              'taimeris
 THEN  RESET           O0.1            'Lampina
       RESET           O0.2
       RESET           O0.3
       JMP TO 1
STEP 2a
 IF                    I0.1
 THEN  SET             O0.3
       SET             T1              'taimeris
STEP 7
 IF             N      T1              'taimeris
 THEN  SET             O0.2
       SET             T1              'taimeris
STEP 8
 IF             N      T1              'taimeris
 THEN  SET             O0.1            'Lampina
       SET             T1              'taimeris
STEP 9
 IF             N      T1              'taimeris
 THEN  RESET           O0.1            'Lampina
       RESET           O0.2
       RESET           O0.3
       JMP TO 1
```

11. Piemērs

Izveidosim programmu, kas izpilda ciklisku lampiņu iedegšanos un nodzišanu atpakaļejošā secībā.

```
STEP 1
 IF                    NOP
 THEN  SET             O0.1            'Lampina
       LOAD            V30
         TO            TP1
       SET             T1              'taimeris
STEP 2
 IF            N       T1              'taimeris
 THEN  SET             O0.2
       SET             T1              'taimeris
STEP 3
 IF            N       T1              'taimeris
 THEN  SET             O0.3
       SET             T1              'taimeris
STEP 4
 IF            N       T1              'taimeris
 THEN  RESET           O0.3
       SET             T1              'taimeris
STEP 5
 IF            N       T1              'taimeris
 THEN  RESET           O0.2
       SET             T1              'taimeris
STEP 6
 IF            N       T1              'taimeris
 THEN  RESET           O0.1            'Lampina
       SET             T1              'taimeris
STEP 7
 IF            N       T1              'taimeris
 THEN
       JMP TO 1
```

Pēc ielādēšanas kontrollerī, programma tiek automātiski palaista. Tā uzreiz turpina strādāt, jo programma sākas ar operatoru NOP (IF NOP THEN – ja nav veikta operācija, tad). Vispirms secīgi iedegas lampiņas 1-2-3, bet nodziest secībā 3-2-1. Programma darbojas nepārtrauktā ciklā.

12. Piemērs

Izveidosim programmu, kas izpilda lampiņu iedegšanos un nodzišanu pēc konkrētiem laika intervāliem, izmantojot taimera tekošo vārdu TW1.

```
STEP 0
 IF                    NOP
 THEN  LOAD            V1000
         TO            TP1             'Taimeris1
STEP 1
 IF                    I0.0            'Ieslegt
 THEN  SET             T1
       SET             O0.0            'Lampina0
STEP 2
 IF              (     TW1
         =             V900      )
 THEN  RESET           O0.0            'Lampina0
       SET             O0.1            'Lampina1
```

```
STEP 3
 IF               (    TW1
          =            V700      )
 THEN  RESET           O0.1           'Lampina1
       SET             O0.2           'Lampina2
STEP 4
 IF               (    TW1
          =            V400      )
 THEN  RESET           O0.2           'Lampina2
       SET             O0.3           'Lampina3
STEP 5
 IF             N      T1
 THEN  RESET           O0.3           'Lampina3
       JMP TO 0
```

Sākotnēji tiek iedegta nultā lampiņa, kas nodziest pēc 1sekundes, tad tiek iedegta pirmā lampiņa, kas deg 2 sekundes, tai nodziestot iedegas otrā lampiņa uz 3 sekundēm, treša lampiņa deg 6 sekundes.

13. Piemērs

Izveidosim programmu, kas izpilda 3 lampiņu ciklisku iedegšanos un nodzišanu 3 reizes.

```
STEP 0
 IF                    NOP
 THEN
       LOAD            V4
         TO            CP1            'Skaititajs
       SET             C1             'Skaititajs
       INC             C1             'Skaititajs
STEP 1
 IF             N      C1             'Skaititajs
 THEN  JMP TO 5
 OTHRW
       SET             O0.1           'Lampina
       LOAD            V50
         TO            TP1
       SET             T1             'taimeris
STEP 2
 IF             N      T1             'taimeris
 THEN  RESET           O0.1           'Lampina
       SET             O0.2
       SET             T1             'taimeris
STEP 3
 IF             N      T1             'taimeris
 THEN  RESET           O0.2
       SET             O0.3
       SET             T1             'taimeris
STEP 4
 IF             N      T1             'taimeris
 THEN  RESET           O0.3
       INC             C1             'Skaititajs
       JMP TO 1
STEP 5
```

```
IF                    NOP
THEN
      SET             O0.4            'Lapm4
      RESET           O0.4            'Lapm4
```

Sākotnēji tiek iedarbināts skaitītājs, kas skaitīs, lai lampiņas nomirgotu tikai 3 reizes. Nākamajā solī tiek testēts vai lampiņām vēl ir nepieciešams iedegties un ja ir, tad veic vienu ciklu un palielina skaitītāja vērtību par vienu vienību. Ja skatītājs ir sasniedzis savu maksimālo vērtību, nomirgo ceturtā lampiņa un programma beidz darbu.

7.3. Pneimatiskā manipulatora vadība

PLC programmēšanas iemaņu apguvei visefektīvāk un interesantāk ir izmantot reālas piedziņas izpildiekārtas, tas ļauj izprast iekārtas vadības programmēšanas specifiku. Pneimatiskas piedziņas manipulators ir darbojoša laboratorijas iekārta, kas aprīkota ar FC-20 kompakto PLC (skat. 7.10. att.).

Tabula 7.1

Kontrollera izeju un manipulatora izpildierīču specifikācija

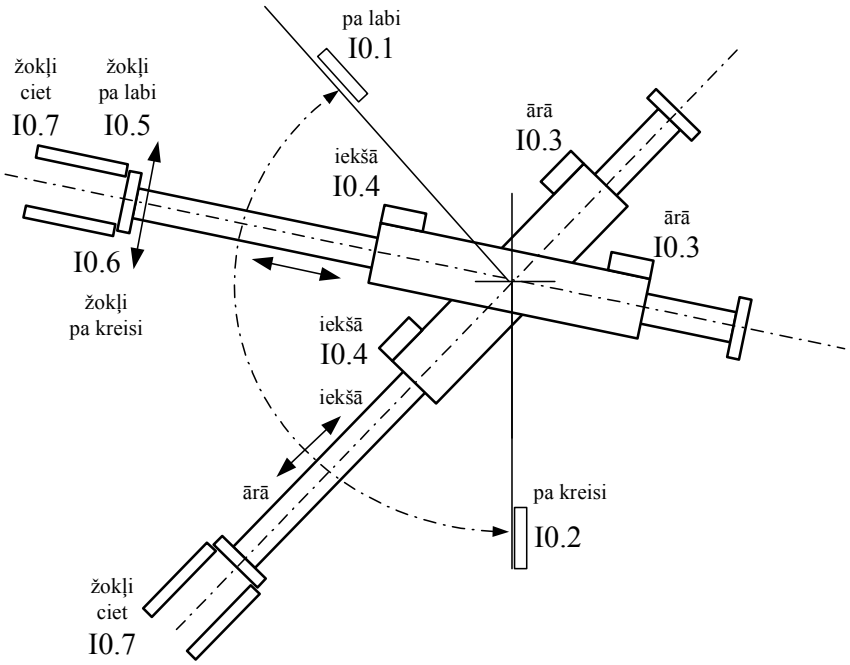
Izeja	Izpildāmā darbība
o0.0	paceļ uz augšu
o0.1	pagrieziens pa labi
o0.2	pagrieziens pa kreisi
o0.3	bīda uz āru
o0.4	bīda uz iekšu
o0.5	bīda galvu pa labi
o0.6	bīda galvu pa kreisi
o0.7	atver žokļus

Tabula 7.2

Kontrollera ieeju un manipulatora sensoru specifikācija

Ieeja	Pozīcijas devējs
i0.0	brīvs
i0.1	max stāvoklis pa labi
i0.2	max stāvoklis pa kreisi

i0.3	1 un 2 izbīdīti ārā (virknes slēg.)
i0.4	1 un 2 iebīdīts iekšā (virknes slēg.)
i0.5	1 galva nobīdīta pa labi
i0.6	1 galva nobīdīta pa kreisi
i0.7	1 žokļi aizvērti
i1.0	"Starts" (brīvs)
i1.1	"Stop" (brīvs)
i1.2	pacelts augšā
i1.3	brīvs



7.10. att. **Galaslēdžu izvietojums uz pneimatiskā manipulatora**

14. Piemērs

Izveidosim vienkāršu konveijera programmu.

```
STEP 0
 IF                    NOP
 THEN
       SET             O0.4            'biida uz iekshu
```

```
       SET            O0.5          'galva
       SET            O0.2          'pa kreisi
       RESET          O0.7          'atver zoklus
       RESET          O0.0          'Pacelj
STEP 1
 IF                   I1.0          'start
 THEN  JMP TO 2
STEP 2
 IF                   I0.4          'iebidits ieksha
       AND            I0.5          'zokli pa labi
       AND            I0.2          'pagriezts pa kreisi
 THEN  RESET          O0.4          'biida uz iekshu
       SET            O0.7          'atver zoklus
       SET            O0.3          'biida uz aaru

STEP 21
 IF            N      I0.7          'zhokli aizveerti
       AND            I0.3          'izbiditi ara
 THEN  RESET          O0.7          'atver zoklus
       RESET          O0.3          'biida uz aaru
       SET            O0.4          'biida uz iekshu

STEP 24
 IF                   I0.4          'iebidits ieksha
 THEN  RESET          O0.2          'pa kreisi
       SET            O0.1          'pa labi

STEP 25
 IF                   I0.1
 THEN  RESET          O0.4          'biida uz iekshu
       SET            O0.3          'biida uz aaru
STEP 26
 IF                   I0.3          'izbiditi ara
 THEN  SET            O0.7          'atver zoklus

STEP 27
 IF                   I0.3          'izbiditi ara
 THEN  RESET          O0.3          'biida uz aaru
       SET            O0.4          'biida uz iekshu

STEP 28
 IF                   I0.4          'iebidits ieksha
 THEN  SET            O0.2          'pa kreisi



STEP 29
 IF                   I0.2          'pagriezts pa kreisi
 THEN  JMP TO 0
```

Programmas sākumā, kad netiek veiktas darbības, manipulators ieņem sākuma stāvokli. Pēc palaišanas pogas nospiešanas manipulators atver žokļus un izbīdās uz āru. Sasniedzot gala stāvokli, žokļi tiek aizvērti un manipulators iebīdās uz iekšu, pēc iebīdīšanās pagriežas pa labi. Sasniedzot gala stāvokli izbīdās uz āru un atver žokļus.

Tad iebīdās uz iekšu, pagriežas pa kreisi un aizver žokļus. Programma atgriežas sākuma stāvoklī.

15. Piemērs

Izveidosim programmu, kas demonstrē skaitītāja vārda CW1 izmantošanu, darbība tiek veikta noteiktu skaitu reižu.

```
STEP 0
 IF                    NOP
 THEN  LOAD            V10
         TO            CP1             'Counter1
       SET             C1
       SET             O0.2            'pa kreisi
STEP 02
 IF                    I0.2            'pagriezts pa kreisi
 THEN
       RESET           O0.2            'pa kreisi
       SET             O0.1            'pa labi
       LOAD            V5
         TO            FW0
STEP 1
 IF                    I0.1
 THEN  LOAD            V50
         TO            TP1             'taaimeris1
       SET             T1
STEP 11
 IF             N      T1
 THEN
       RESET           O0.1            'pa labi
       INC             C1
       SET             O0.2            'pa kreisi
STEP 2
 IF               (    CW1
         =             FW0        )
 THEN  SET             O0.7            'atver zoklus
 OTHRW JMP TO 02
```

Sākotnēji manipulators ieņem sākuma stāvokli un tiek palaists skaitītājs. Ja ir ieņemts sākuma stāvoklis, tad manipulators pagriežas pa labi un tiek izveidota atmiņas FW0 vārda vērtība vienāda ar 5, kura tiks izmantota, lai salīdzinātu ar skaitītāja vārdu CW1. Katra gala stāvokļa beigās skaitītāja vērtība tiek palielināta un tā vārds tiek salīdzināts ar atmiņas FW0 vērtību. Ja vērtības sakrīt, tad tiek atvērti žokļi un programma beidz darbu. Ja vērtības nesakrīt, tad programma atgriežas sākumā un izpilda pagriešanos atkārtoti, līdz skaitītāja un atmiņas vērtības sakrīt.

16. Piemērs

Izveidosim programmu, kas izpilda pakāpenisku žokļu atvēršanās un aizvēršanās ātruma pieaugumu.

```
STEP 0
 IF                    NOP
 THEN  SET             O0.2            'pa kreisi
       SET             O0.3            'biida uz aaru
       RESET           O0.7            'atver zoklus
       LOAD            V100
         TO            TP1             'taaimeris1
       LOAD            V100
         TO            FW0
       LOAD            V10
         TO            FW1             'atminja 1
STEP 1
 IF                    I1.0            'start
 THEN  SET             T1
STEP 2
 IF                    I0.7            'zhokli aizveerti
       AND     N       T1
 THEN  LOAD            FW0
         TO            TP1             'taaimeris1
       SET             O0.7            'atver zoklus
       SET             T1

STEP 3
 IF            N       I0.7            'zhokli aizveerti
       AND     N       T1
 THEN  LOAD      (     FW0
         -             FW1       )     'atminja 1
         TO            FW0
       RESET           O0.7            'atver zoklus
       SET             T1
STEP 4
 IF              (     FW0
         =             V0        )
 THEN
       LOAD            FW1             'atminja 1
         TO            FW0
       JMP TO 2
 OTHRW JMP TO 2
```

Sākotnēji tiek ieņemts sākuma stāvoklis un ielādētas nepieciešamās vērtības taimerī T1 un atmiņās FW0 un FW1. Nospiežot palaišanas pogu, tiek palaists taimeris. Katra cikla sākumā taimerī tiek ielādēta atmiņas FW0 vērtība, kas katra cikla beigās tiek samazināta par FW1 vērtību, 10. Katru reizi ielādējot taimerī jauno vērtību, žokļi atveras un aizveras arvien ātrāk. Kad FW0 ir sasniedzis minimālo stāvokli, tad žokļi tiek nepārtraukti virināti ar konstantu laika intervālu.

17. Piemērs

Izveidosim programmu, kur notiek manipulatora pacelšanas, iebīdīšanas un izbīdīšanas kustības.

```
STEP 0
 IF                    I1.0          'starts
 THEN  SET             O0.0          'uz augshu
STEP 1
 IF                    I1.2          'pacelts uz augshu
 THEN
       SET             O0.4          'iebida uz iekshu
STEP 2
 IF                    I0.4          'iebiidiits uz iekshu
 THEN  SET             O0.7          'atver zoklus
STEP 4
 IF             N      I0.7          'zokli atverti
 THEN  RESET           O0.4          'iebida uz iekshu
       SET             O0.3          'izbida uz aru
STEP 5
 IF                    I0.3          'izbidits uz aru
 THEN  RESET           O0.3          'izbida uz aru
       RESET           O0.0          'uz augshu
       RESET           O0.7          'atver zoklus
       JMP TO 0
```

Pēc palaišanas pogas nospiešanas tiek veikta manipulatora pacelšana uz augšu. Kad ir sasniegts gala stāvoklis, tad tiek iebīdītas rokas un atvērti žokļi. Tad manipulatora rokas tiek izbīdītas uz āru un manipulators nolaists lejā. Beigās tiek aizvērti žokļi un programma atgriežas sākuma stāvoklī.

18. Piemērs

Izveidosim programmu, kur notiek manipulatora pagriešanās un pacelšanās uz augšu.

```
STEP 1
 IF                    I0.0          'Start
 THEN  SET             O0.1          'pa labi
STEP 2
 IF                    I0.1          'Pa labi
 THEN  RESET           O0.1          'pa labi
       SET             O0.7          'atver zoklus
STEP 22
 IF             N      I0.7          'zhokli aizverti
 THEN
       RESET           O0.7          'atver zoklus
```

```
STEP 23
 IF                    I0.7          'zhokli aizverti
 THEN
       SET             O0.7          'atver zoklus
       SET             O0.2          'pa kreisi
STEP 3
 IF                    I0.2          'Pa kreisi
 THEN  RESET           O0.2          'pa kreisi
       SET             O0.0          'Uz augsu
STEP 4
 IF                    I1.2          'Pacelts
 THEN  RESET           O0.0          'Uz augsu
       RESET           O0.7          'atver zoklus
       JMP TO 1
```

Sākotnēji manipulators tiek pagriezts pa labi un 2 reizes atver žokļus. Tad pagriežas atpakaļ, paceļas uz augšu un aizver žokļus.

Dotie programmēšanas piemēri nav veidoti, kā tehnoloģiska procesa sastāvdaļas, tie parāda tikai programmēšanas valodas elementu pielietojumu. Konkrētā praktiskajā pielietojumā, ir jāizprojektē precīzs tehnoloģiskā procesa vadības algoritms un tikai tad var programmēt kontrollerus.

Izmantotā literatūra

1. D.Collins and E.Lane Programmable controllers. A practical guide. McGraw-Hill Book company, 1995. – 169 p.
2. R.Bliesener and oth. Programmable logic controllers. Festo Didactic KG, Esslingen, 2002. – 206 p.
3. Trumpas FST 4 programavimo paketo aprašymas. KTU Kaunas, 2002. – 25 p.
4. Http://www.festo.lt (25.03.2007)
5. Vinck S. Introduction in FST programming for the FEC. STL language. © FESTO Belgium 1998. – 51 p.
6. Batten, G. L. Programmable controllers: hardware, software, and applications- New York etc.: McGraw-Hill, 1994. – 281 p.
7. Hooper, J. F. Introduction to PLCs. - Carolina Academic Press, 2004. – 120 p.
8. Thomas A. H. Programmable controllers, Third edition – USA : Instrumentation, Systems, and Automation Society – ISA, 2001. – 352 p.
9. Как выбрать программируемый логический контроллер http://magazine.stankin.ru/arch/n_10/index.shtml (10.03.2006).
10. Cox, R. Technician's Guide to Programmable Controllers. - Thomson Delmar Learning, 2000. – 384 p.