

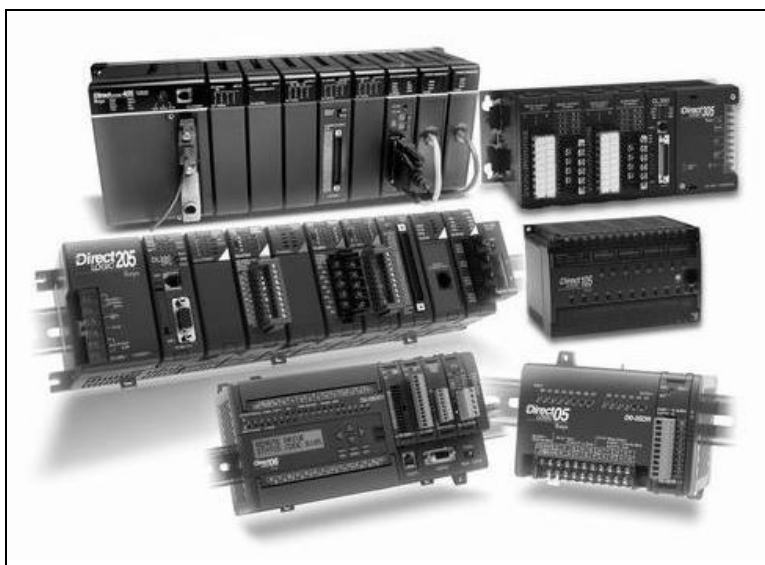


УНИВЕРЗИТЕТ “СВ. КИРИЛ И МЕТОДИЈ”

МФС

МАШИНСКИ ФАКУЛТЕТ - СКОПЈЕ

# ПРОГРАМИБИЛНО МЕМОРИСКО УПРАВУВАЊЕ ПРОГРАМИБИЛНИ ЛОГИЧКИ КОНТРОЛЕРИ



- интерна скрипта за студентите на МФС -

проф. д-р Атанаско Тунески  
асс. м-р Дарко Бабунски

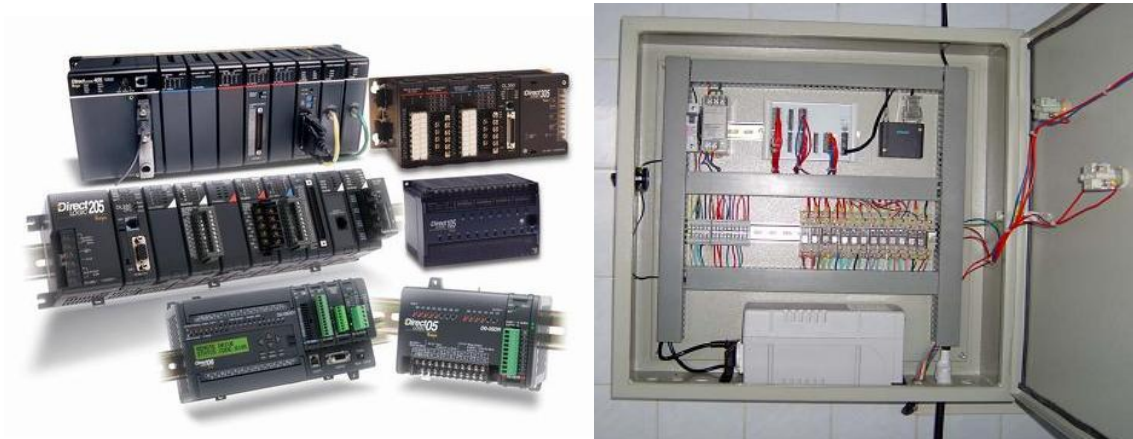
СКОПЈЕ 2009

## 1. Вовед

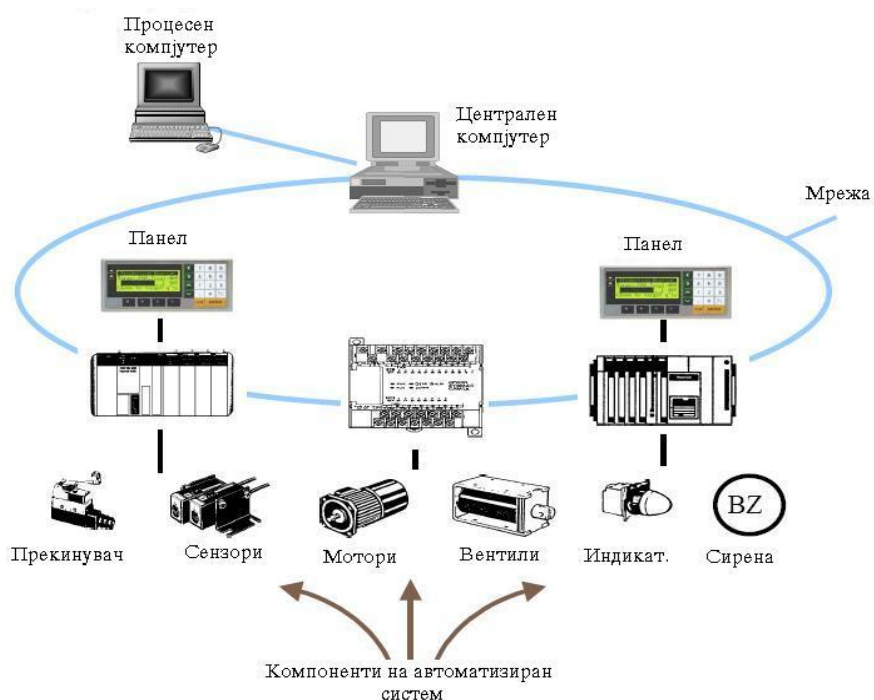
Системот за управување во машинството и електротехниката го сочинуваат збир на уреди и опрема кои обезбедуваат стабилност, точност и елиминација на штетните состојби во производните процеси. Системот за управување може да биде со различен облик и имплементација, од енергетски постројки до производствени машини. Системите на управување се развивале со текот на времето. Во раниот период на развојот самите луѓе ги вршеле управувачките задачи. Кон крајот на 60-тите години од минатиот век, системите за управување биле базирани на примери на релејна логика, врз база на релативно едноставни логички алгоритми. Напредокот на технологијата во изработката на микропроцесорите во тоа време, довело до револуција во системите на управување. Се појавила идеја за изработка на електронско-микропроцесорски управувачки уреди кои би можеле едноставно да се репрограмираат во случај на промена во управувачките задачи. Таквите уреди се наречени *Програмибилни Логички Контролери* (**Programmable Logic Controllers**) или скратено PLC. Понатамошниот развој на овие уреди бил многу брз, бидејќи имале голем број на предности во однос на логиката заснована на примената на релеата, бидејќи немаат механички задвижувачки делови, пофлексибилни се поради можноста за програмирање, димензиите се многу помали, имаат помала сопствена потрошувачка. Според здружението на производители на електрична опрема (The National Electrical Association–NEMA) програмибилните логички контролери, дефинирани се како “Дигитален електронски уред, кој користи програмибилна меморија за запишување на наредби со чија помош се извршува некоја специфична функција, како што се логичките функции, секвенцирање, пребројување, мерење на времето, пресметки, управување на различни машини и процеси.

PLC како индустриски компјутер, со самиот свој дизајн, предвиден е за примена во непосредно опкружување на процесот со кој што управува, така што отпорен е на различни неповолни влијанија, прашина, влага, висока температура, вибрации и електромагнетни пречки, така што често се применува за решавање на децентрализираните управувачки задачи на самото место на управување, каде што се поврзува преку влезови и излези со уредите како што се: операторски панели, мотори, сензори, прекинувачи, вентили и сл. PLC како и секој сметач има оперативен систем, кој секако има многу помалку можности за разлика од оперативни системи за општа намена. Можно е да се изведе поврзување на програмибилните логички контролери (PLC) и евентуално со централен компјутер или друг компјутер за решавање на сложени управувачки задачи или едноставни аквизации на податоци и управување од далечина. Можностите за комуникација помеѓу PLC уредите се толку големи што овозможуваат висок степен на искористување и координација на процесите, како и голема флексибилност во реализација на управувачките процеси, така да можноста за комуникација како и флексибилноста, претставуваат главни предности за решавање со PLC уредите.

PLC е елемент од автоматизиран систем, кој врз основа на прифатените влезни сигнали од влезните уреди, по определен програм, формира излезни сигнали преку кои управува со излезните уреди. Во автоматизиран систем, PLC контролерот е центар на управување, со извршување на програмата сместена во програмската меморија, PLC непрекинато ја набљудува состојбата на системот преку влезните уреди. Врз база на логика имплементирана во програмот на PLC, одредува кои акции треба да се извршат врз излезните уреди. За управување на сложени процеси, постои можност да се поврзат повеќе PLC-а меѓусебе или со централен компјутер.



Слика 1.1: Изглед на PLC уредите



Слика 1.2: PLC во систем за управување

Според бројот на влезно/излезни приклучоци, PLC уредите во принцип може да се поделат на *микро* со максимум до 32, мали до 256, средни до 1024 и *големи* PLC-а преку 1024 влезно/излезни приклучоци. Со зголемување на бројот на приклучоци, мора да се зголеми и брзината на процесорот како и количината на меморијата, а со самото тоа растат сложеноста и цената на самиот уред.

Предности на PLC системите во однос на релејните системи се:

- Сигурност - нема механички движечки делови имат голема отпорност на прашина, влага, високи температури, вибрации, електромагнетни влијанија, високи фреквенции поради тоа што штапаната плочка на PLC е направен од специјален материјал ROGER како изолатор кој се става помеѓу слојевите

(материјалот е врз база на керамика), ако дојде до исклучување на напојувањето ништо не се менува а ако напојувањето се вклучи PLC продолжува со работа. Грешки во ожичувањето се сведуваат на минимум а ожичувањето се сведува на очичување на влезот и излезот на PLC.

- Адаптивност - кога ќе се напише и тестира PLC програма за управување без проблеми може да се примени во друг PLC.
- Флексибилност - за промена на програма потребно е многу малку време. Изведувачите на управувачкиот систем можат без проблем да му ја испратат на корисникот изменетата програма преку дискета или друг мемориски уред, без да испраќа техничари за одржување. Корисникот може едноставно да ја пренесе програмата во PLC и да изврши мали модификации.
- Напредна функционалност - програмските пакети кој се користат за програмирање на PLC им нудат на проектантите бројни можности што е невозможно да се изведат со релејно управување.
- Комуникација - PLC може да се врзе со други уреди, овозможува аквизиција на податоци од другите уреди и обработка на аквизираните податоци.
- Дијагностика - PLC нуди брзо и едноставно отстранување софтверски и хардверски грешки во управувачките системи.

## 2. Општо за програмибилните логички контролери (PLC)

Систем кој што се автоматизира, односно систем на кој што сакаме да примениме автоматско управување, се нарекува **објект на управување**. Работата на објектот на управување константно се прати со помош на влезни уреди (сензори) кои му даваат информации на PLC-то за случувањата во системот. Како одговор на тоа PLC-то праќа сигнал на надворешните извршни елементи, кои всушност ја контролираат работата на системот, на начин на кој што програмерот го програмира системот. Програмерот, PLC-то го програмира врз основа на критериумите дефинирани со технолошките задачи. Програмот се испишува во наменскиот програмски јазик, каде секој производител го дава со својот PLC уред, а кој претставува комбинација од програмски едитори, компајлери и комуникациониот софтвер. Во едиторот се испишува програмот, пратејќи го редоследот на операциите на управување, а потоа се проверува неговата синтакса и врши компајлирање. Ако е сè во ред, со помош на комуникациска врска софтверот се праќа во меморијата на PLC уредот, каде се сместува и покренува. Влезните и излезните уреди, кои се поврзуваат со PLC контролерот, оптимално се одбираат врз основа на барањата и поставените критериуми дефинирани во технолошката задача која треба да ја задоволат. Влезните уреди се: прекинувачи, сензори и давачи. Излезните уреди можат да бидат соленоиди, релеа, електромагнетни вентили, мотори, контактори како и уреди за светлосна и звучна сигнализација.

## 2.1. Влезни уреди

Влезните уреди чии сигнали ги прифаќа PLC уредот, можат да бидат различни. Од типот на сигналот кој го даваат на својот излез може да се поделат на дигитални и аналогни, додека аналогните можат да бидат фреквентни (во херци), напонски (0-10V) и струјни (0-20 mA или најчесто 4-20 mA). На влезните сигнали мора понатаму да се изврши прилагодување со соодветните влезни модули.



Пневматски граничен прекинувач



Фотосензор



Енкодер



Индукциски прекинувач

Слика 2.1: Типични влезни уреди

## 2.2. Излезни уреди

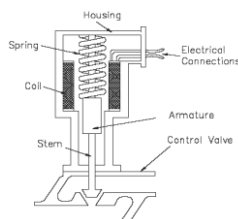
Излезните уреди се уреди кој се управувани од PLC на основа на програмот и состојбата на влезите. Излезните уреди исто и како влезните се делат на дигитални и аналогни уреди при што аналогните ги делиме на фреквентни (во херци), напонски (0-10V) и струјни (0-20 mA или најчесто 4-20 mA), со што со помош на PLC контролерот се прилагодуваат на потребни напонски и струјни нивоа. Излезите на PLC се обично галванско изолирани контакти, кои можат да се изведат како транзисторски (за помали излезни моќности) и релејни (за поголеми излезни моќности).



Мотор



Грејач



Соленоид



Светилка



Дисплеј



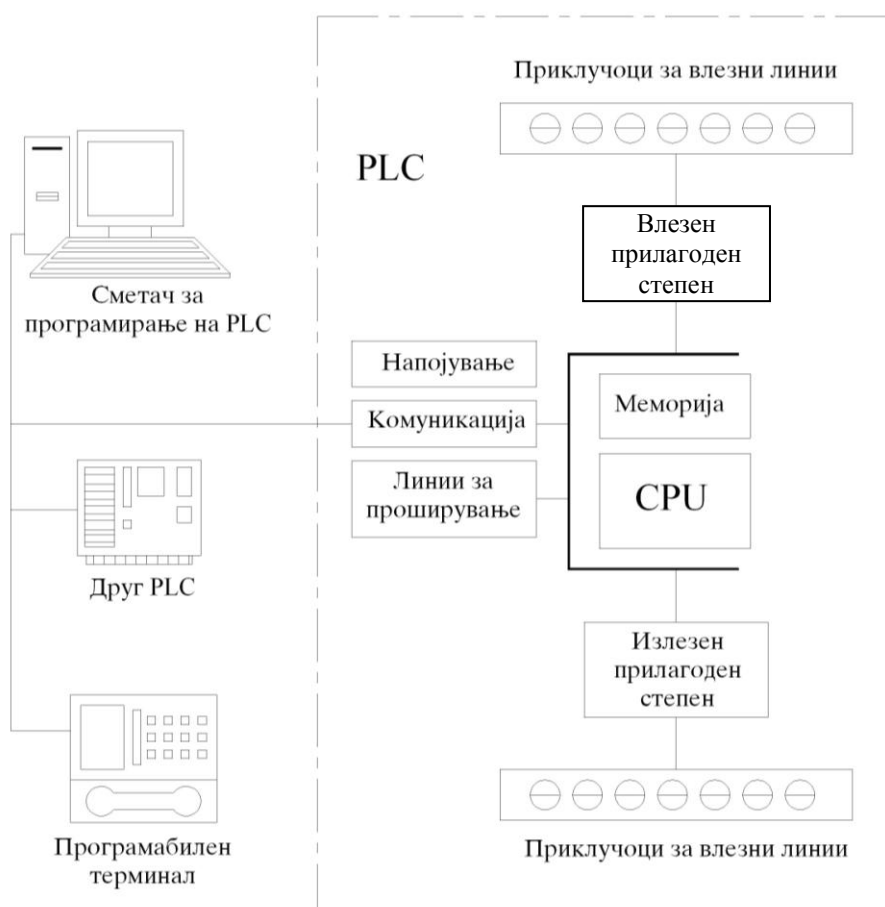
Релеен контактор

Слика 2.2: Типични излезни уреди

### 2.3. Основни делови на PLC и нивна функција

За да се објасни начинот на работа на PLC уредите потребно е да видиме од кој основни делови е составен и нивната функција. Сите PLC уреди од микро до најголеми имаат во принцип иста хардверска структура, исти цели на функционирање.

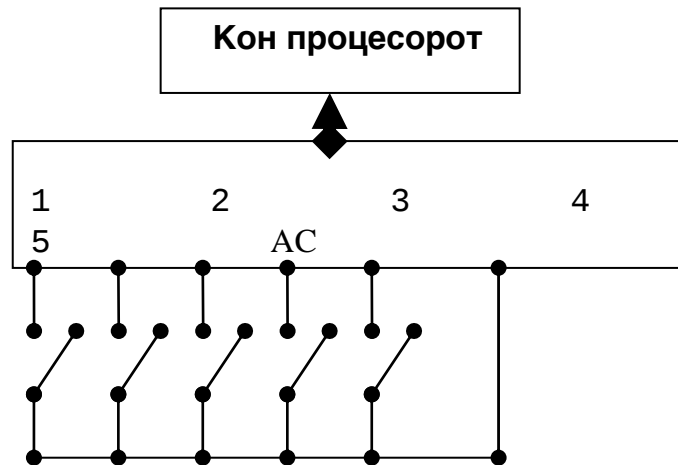
- Влезен модул (дигитален, аналоген влез)
- Излезен модул (дигитален, аналоген излез)
- CPU т.е централна процесорска единица
- Мемориски блок
- Мрежен модул за напојување
- Модул за проширување
- Комуникациско поврзување



Слика 2.3: Блок дијаграм на PLC

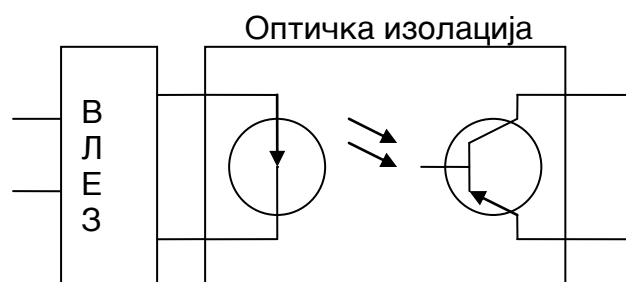
## 2.4. Влезен модул

Влезниот модул е место каде што се приклучуваат влезните уреди односно ги прима влезните сигнали и место каде што влезните сигнали се трансформираат во сигнали разбирливи за централно процесорската единица. Карактеристични влезни уреди се тастери, прекинувачи, сензори, енкодери, термостати и други. Влезните сигнали можат да бидат дигитални и аналогни.



Слика 2.4: Типично поврзување на влезен модул

Задачата на влезниот модул е да го претвори влезниот сигнал во облик 1 или 0 состојба која што е потребна за процесот. Влезните модули имаат канали во кој има индикаторско светло кое свети кога некој влез е вклучен или не свети кога влезот е исклучен. Светлината ја прима фото транзистор кој дава сигнал 1 кога свети и 0 кога не свети. На следната слика ни е прикажано како изгледа тоа конвертирање. Како индикаторско светло се користи LED диода. Фото транзисторот заедно со LED диодата се изолираат оптички во едно залиено куќиште.



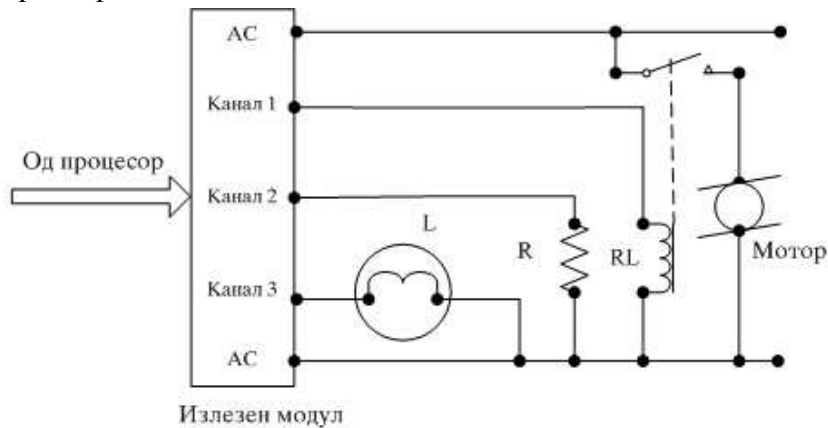
Слика 2.5: Конвертирање на сигнал

Аналогните влезови можат да бидат струјни или напонски. Аналогните напонски сигнали можат да бидат униполарни и биполарни. Униполарните влезови прифаќаат напон од еден поларитет, на пример од 0 до +10 V. Биполарните напонски влезови можат да прифатат напон од два поларитета од -10 V до +10 V.

Аналогните струјни влезови најчесто прифаќаат струја од опсег од 4mA до 20mA.

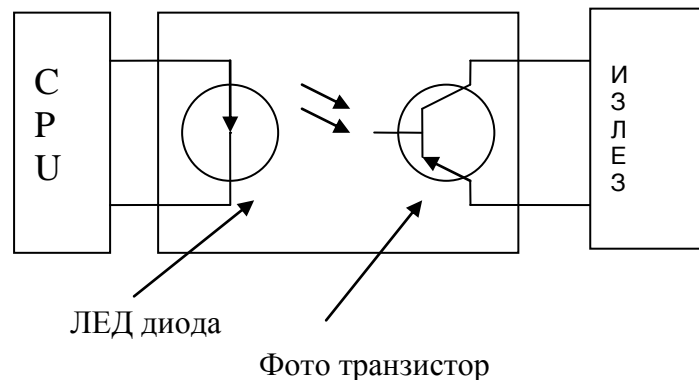
## 2.5. Излезен модул

Излезните модули се места каде што се поврзуваат излезните уреди и места каде што се врши трансформација на логички 1 или 0 што ги испраќа процесорот во дигитален или аналоген сигнал. На излезот најчесто се поврзуваат: релеја, склопки, мотори, светилки, пневматски распоредници и сл.



ИЗЛЕЗЕН МОДУЛ  
Слика 2.6: Типично поврзување на излезен модул

Местото каде што се врши трансформацијата на сигналот мора да биде галвански одвоено. Централно процесорската единица го носи сигналот на LED диодата со што ја вклучува или исклучува (свети, не свети), светлината го побудува фототранзисторот кој активира излезен уред, а најчесто реле затоа што тоа може да изврши прекинување на големи напонски и струјни сигнали.



Слика 2.7: Конвертирање на сигнал

## 2.6. Централна процесорска единица CPU

Централна процесорската единица е микропроцесорски систем кој содржи системска меморија и служи како управувачка единица. CPU е мозокот на PLC. Се грижи за комуникација, меѓусебно поврзување со останатите делови на контролерот, извршување на зададена програма и поставување на излез.



## 2.7. Мемориски блок

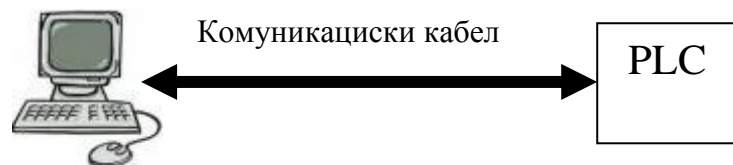
РАМ меморијата е привремена меморија која се користи додека вршине тестирање и пресметки. Оваа меморија овозможува бришење, корегирање на грешка која се учила при тестирање. Обично се напојува со посебна батерија за да не дојде до бришење на податоците при прекин на напојувањето. Кога програмот ќе биде комплетно завршен тој се сместува во РОМ меморија. Во РОМ меморијата програмот се сместува трајно со што не може да се избрише ако дојде до прекин на напојувањето.

## 2.8. Мрежен модул за напојување

Мрежниот модул за напојување е најгабаритниот и најтешкиот дел во PLC. Не е осетлив на пречки и кратки испади на мрежниот напон. Стандардни влезови за напојување на PLC уредите се: 110/220VAC и 24VDC.

## 2.9. Комуникациско поврзување

Комуникацискиот модул извршува значајна работа. Прво и основно врши поврзување на PLC со компјутерот на кој се проектира управувачката програма и се испраќа во PLC.

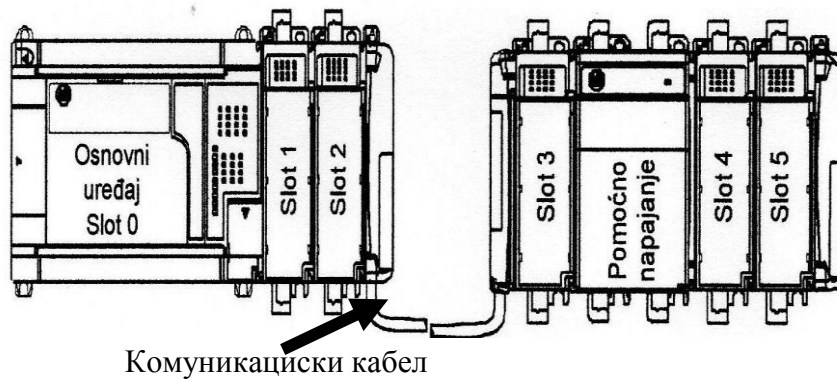


Други можности за комуникација е поврзување со други уреди. Обично сите PLC уреди имаат вграден сериски порт за комуникација RS232. други најчесто применувани комуникациски поврзувања се Ethernet, Canbus, Profibus, RS-422, RS-485. Производителите на уреди имаат стандардизирано свои протоколи низ кој се извршува преносот на податоци.

## 2.10. Модули за проширување

Во основа програмибилните логички контролери имаат одреден број на влезови и излези. Кога во процесот ќе ни бидат потребни повеќе влезови и излези од колку што има основниот уред се користат модули за проширување (slot). Модулот за проширување е посебен уред кој се спојува со PLC и има додатни влезови и излези. PLC секогаш можат да се прошируваат без да се набавуваат нови. Најчесто модулите за проширување се продаваат како модули за дигитален влез или излез, и модули за аналоген влез или излез. Модулите се напојуваат со електрична енергија од самиот уред или тие можат да се напојуваат и од надворешен извор на електрична енергија. Модулите за проширување можат да бидат и одалечени од основниот уред со поврзување на комуникациски кабли. Бројот на модули за спојување зависи од производителот. На сликата е прикажан PLC со модул за проширување на влезовите и излезите. Дополнителниот модул може да се напојува од сопствено напојување или да

го користи постојното напојување на PLC. Препорачливо е PLC и дополнителниот модул да користат заедничко напојување.



Слика 2.8: Дополнителен влезно-излезен модул

## 2.11. Работа на PLC

1. Прв чекор - прво PLC ги проверува сите влезни линии за да утврди која има статус ON (вклучено) која OFF (исклучено). Регистрираните податоци ги пренесува во влезниот мемориски регистар на процесорската единица.
2. Втор чекор - се земаат регистрираните влезни податоци и се извршува обработка на програмата, потоа обработените податоци се регистрираат во излезниот мемориски регистар.
3. Трет чекор - податоците од излезниот мемориски регистар се пренесуваат кон излезите на PLC.
4. Четврт чекор - се врши размена на податоци со уредите со кои е поврзан PLC.
5. Пети чекор - се врши одржување односно се ажурираат сите интерни часовници и регистри, се контролира исправноста на PLC.

Проверката, дали програмскиот циклус се извршува правилно, се врши со помош на т.н. watch-dog тајмер, кој се повикува во секој скен и претставува основна гаранција за сигурна работа. Ако тоа не се случи, сигнализира грешка во самата програма. На тој начин системот се штити на пример од влез во бесконечен омча. Во зависност од применетиот тип на процесор во PLC-то, влезниот и излезниот скен се извршуваат во милисекунди (0.1 до 3ms), така да циклусот на обработка се повторува од 10 до 100 пати во секунда. Траењето на скенирачкиот циклус на обработка, посебно програмскиот дел, зависи од големината на програмата.

Работа на уредот сликовито е прикажана на слика 2.9.

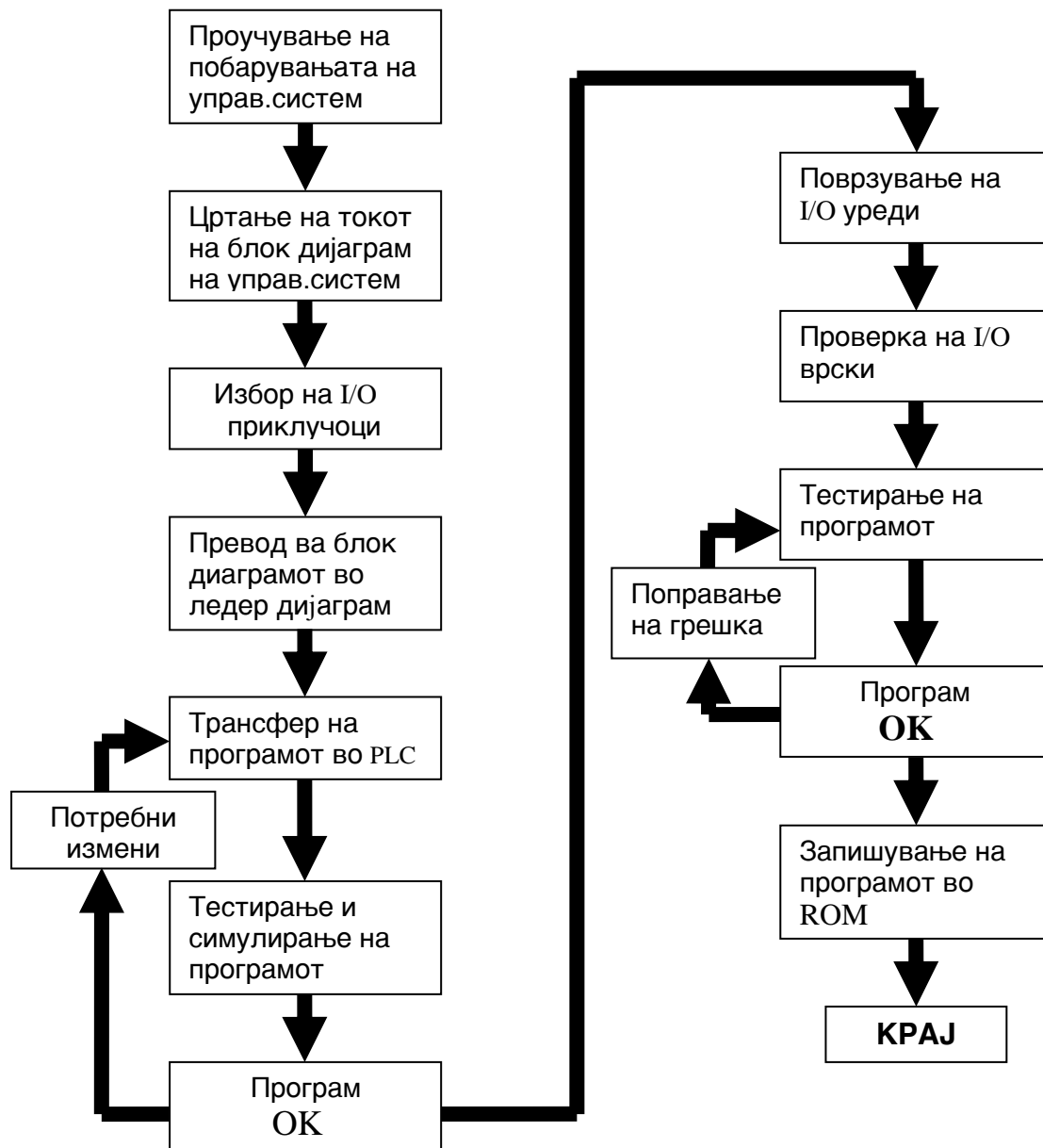


Слика 2.9: Работен циклус на PLC-то



Слика 2.10: Движење на информација во работен циклус на PLC-то

## 2.12. Насоки при програмирање на еден систем управуван со PLC



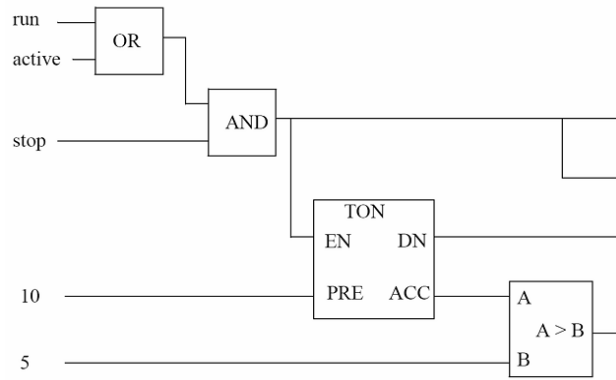
Слика 2.11: Насоки при програмирање

## 2.13. Начини на програмирање на PLC

Програмирањето на PLC всушност представува создавање на соодветна потребна програма со која се врши управување на PLC

Видови на програмски јазици кои се користат за програмирање на PLC се:

- STL - statement list (структурен текст)
- FBT – функциски блок дијаграм
- Ледер (Ladder) дијаграм



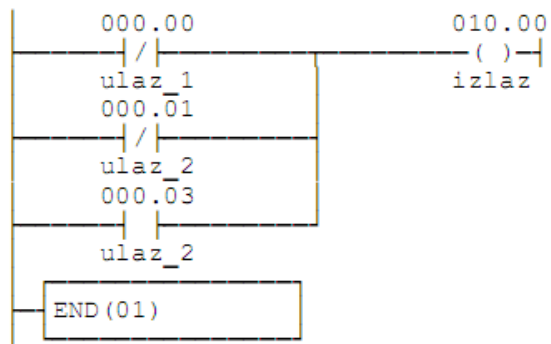
Слика 2.12: Функциски блок дијаграм

```

PROGRAM example
VAR_INPUT
  run : BOOL ;
  stop : BOOL ;
END_VAR
VAR_OUTPUT
  heater : BOOL ;
  fan1 : BOOL ;
  fan2 : BOOL ;
END_VAR
VAR
  active : BOOL ;
  delay : TON ;
END_VAR
active := (run OR active) & stop ;
heater := active ;
delay(EN := active, PRE := 10) ;
IF ( delay.ACC > 5 ) THEN
  fan1 := 1 ;
ELSE
  fan1 := 0 ;
END_IF ;
fan2 := delay.DN ;
END_PROGRAM

```

STL лист (редоследно изведувачки наредби)

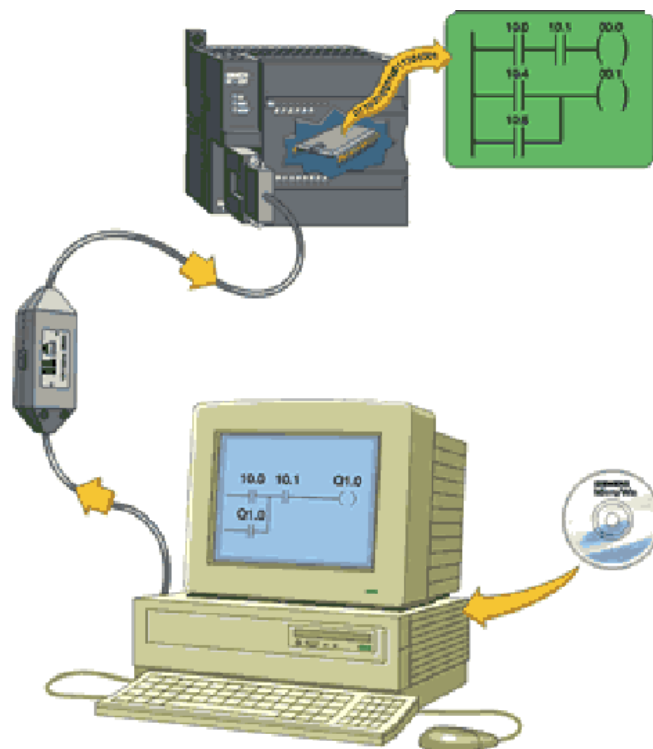


Слика 2.13: Ледер дијаграм

Некои производители нудат и можност за програмирање со помош на BASIC и C, но тие програмски јазици немаат широка застапеност. Секој производител со својот PLC обезбедува софтвер за создавање на програмски код. Софтверот се инсталира на персонален компјутер каде се врши програмирање на соодветната програма. Комуникацијата помеѓу компјутерот и PLC-то најчесто е се извршува преку сериски порт RS-232 и може да биде активна при извршувањето на програмата, со што на мониторот можеме да ја следиме работата на програмата, влезовите и излезите а исто така може и да задаваме наредби преку тастатурата и глушецот.

Програмирањето на PLC-то може да се изврши и преку рачен програмер кој поседува скроман LCD дисплеј со скромна тастатура. Такви уреди се користат за кратки програми или кога сакаме да направиме некоја мала измена во погон. Некои PLC уреди поседуваат на себе мали дисплеји и скромна тастатура што ни овозможува програмирање и кратки програми и некои мали измени на лице место.

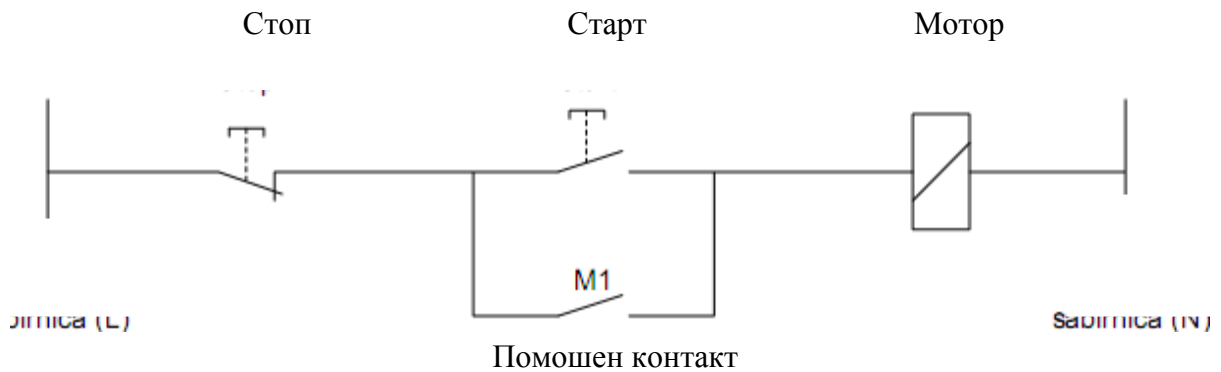
Некои PLC уреди поседуваат и слотови за мемориски картички кои го олеснуваат програмирањето односно измени на програмот во текот на работењето. Доволно е да се исклучи PLC-то да ја замениме мемориската картичка на која однапред е запишана новата програма да го вклучиме PLC-то кој автоматски ја прифаќа новата програма.



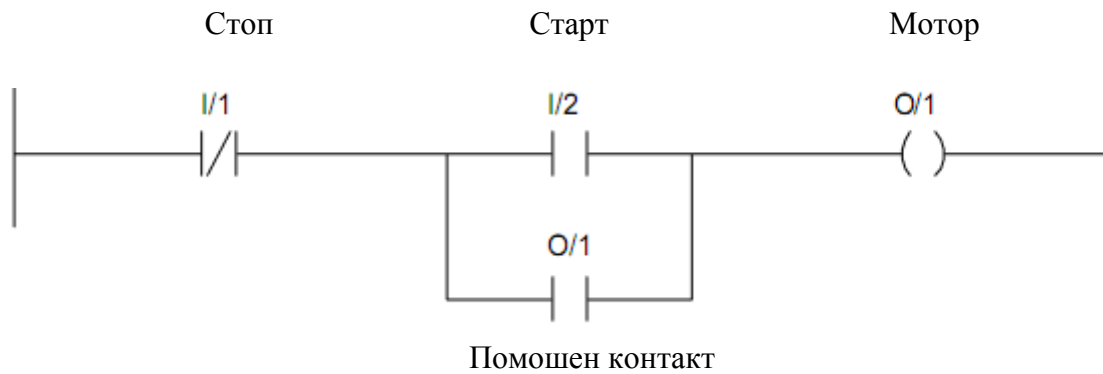
**Слика 2.14:** Комуникација помеѓу PC и PLC

Програмибилните логички контролери најчесто се програмираат во Ледер дијаграм кој е графичко симболично представување на електричните кола. Одбраните симболи на елементите во Ледер програмата изгледаат слично како и шематските симболи на електричните уреди. Кај електричната шема симболите се представени како вистински елементи, додека кај Ледер дијаграмот се користат слични симболи но тие представуваат наредби. Друга разлика е тоа што електричната шема го представува

вистинскиот тек на струјата во електричното коло односно ја прикажува состојбата на контактите (отворен или затворен), додека Ледер дијаграмот е управувачки софтвер кој испитува дали некоја наредба е вистина (1) или неистина (0) и прикажува тек на информација, не на енергија.



Слика 2.15: Електрична шема



Слика 2.16: Ледер дијаграм

### 3. “Unitronics” PLC: хардвер, монтажа, влезови, излези

#### 3.1. Производител

Програмибилниот логички контролер V130-33-TA24 е произведен од фабриката Unitronics во серијата Vision . Оваа фабрика се занимава со произведување и продажба на програмибилни логички контролери кои се применуваат во сите гранки на индустријата. Предности на Unitronics-овите производи е пред сè едноставноста на програмирањето, сигурното и едноставно комуникациско поврзување (Ethernet, MODBUS, CANbus) можност за безжично комуницирање, исто така овозможува бесплатна програмска опрема и релативно пристапни цени.

#### 3.2. Техничка спецификација



Слика 3.1: Unitronics Vision 130 -33 TA24

- 8 дигитални влезови во кој е вклучен еден shaft-encoder
- 2 аналогни /дигитални влезови
- 2 термoeлементи /PT100 / дигитален влез
- 10 транзисторски излези
- 2 аналогни излези
- Графички лед дисплеј бел 128x64 пиксели
- 1 сериски порт RS232/RS485
- 1 порт за Ethernet комуникација
- 1 порт за CANbus комуникација
- Поддржува MODBUS комуникација



## Напојување

Напон на влез	24VDC
Дозволен опсег	20.4VDC до 28.8VDC

## Максимална струјна потрошувачка

pnр влезови	225mA/24VDC
рnr влезови	190mA/24VDC

Вистинската потрошувачка се пресметува на следниот начин: се одземаат струите за секој неупотребен елемент од максималната струјна потрошувачка вредност според:

backlight	Ethernet картичка	Релејни излез (по излез)	Сите аналогни излези V/A
10 mA	35mA	5 mA	48mA/30mA

## Дигитални влезови

Број на влезови	12
Галванска изолација	нема
Номинален влезен напон	24VDC

## Напон на влез

pnр	0-5VDC за Logic '0' / 17-28.8VDC за Logic '1'
рnr	17-28.8VDC за Logic '0' / 0-5VDC за Logic '1'
Струја на влез	3.7mA/24VDC
Импеданса на влез	6.5K $\Omega$
Време на одзив	10mS,кога се употребуваат нор.дигитални влезови
Должина на каблите на влез	до 100 метри , со сноп на жички (лицни)
Резолуција	32-bit
Фреквенција	10kHz максимална

Овој програмибилен логички контролер има од 12 влезови. Сите 12 влезови можат да бидат употребени како дигитални влезови тие можат да бидат групирани со помош на џампер како pnр или рnr. Влезовите 5 и 6 може да функционираат и како дигитални и како аналогни. Влезот 0 може да функционира како брз бројач, и како shaft-encoder или како нормален дигитален влез. Влезот 1 може да биде или како counter reset, нормален дигитален влез, и како shaft-encoder. Влезовите 7-8 и 9-10 можат да функционираат

како дигитални, термоелементи, или РТ 100 влез, влез 11 може да работи како СМ сигнали за РТ 100.

#### Аналогни влезови

Број на влезови	2	
Влезен опсег	0-20mA, 4-20mA	0-10VDC
Влезна импеданса	12.77k $\Omega$ 37 $\Omega$	
Галванска изолација	нема	

#### RTD влезови

Видот на RTD	PT100
Температурен коефициент	0.00385/0.00392
Влезен опсег	-200 до 600°C, 1 до 320 $\Omega$ .
Изолација	нема
Влезна импеданса	>10M $\Omega$
Помошна струја за ПТ100	150 $\mu$ A
Размерна грешка	$\pm$ 0.4%
Линеарна грешка	$\pm$ 0.04%
Статус индикација	да

Аналогната вредност може да прикаже грешка како што е покажано подолу:

Вредност	Можен случај
32767	Сензорот не е поврзан на влез(е во прекин) или вредноста го надминува дозволениот опсег
-32767	Сензорот е во куса врска

#### Термоелементи на влез

Изолација	нема
Метода на претворање	напон во фреквенција
Резолуција	0.1°C/ 0.1°F максимум
Време на претворање	100mS минимум по канал
Импеданса на влез	>10M $\Omega$
Размерна грешка	$\pm$ 0.4%
Линеарна грешка	$\pm$ 0.04%
Време на загревање	карактеристично 1/2час
Статус идикација	да

Мерката на напонот во уредот е во опсег од -5 до 56mV со резолуција од 0.01mV, необработената фреквенција е со резолуција од 14-bits (16384). Опсегот е прикажан на следнава табела:

Тип	Температурен опсег
mV	-5 до 56mV
B	200 до 1820 <sup>0</sup> C (300 до 3276 <sup>0</sup> F)
E	-200 до 750 <sup>0</sup> C (-328 до 1382 <sup>0</sup> F)
J	-200 до 760 <sup>0</sup> C (-328 до 1400 <sup>0</sup> F)
K	-200 до 250 <sup>0</sup> C (-328 до 2282 <sup>0</sup> F)

Тип	Температурен опсег
N	-200 до 1300 <sup>0</sup> C (-328 до 2372 <sup>0</sup> F)
R	0 до 1768 <sup>0</sup> C (32 до 3214 <sup>0</sup> F)
S	0 до 1768 <sup>0</sup> C (32 до 3214 <sup>0</sup> F)
T	-200 до 400 <sup>0</sup> C (-328 до 752 <sup>0</sup> F)

#### Дигитални излези

Број на излези	10 транзистор PNP
Вид на излез	P-MOSFET
Изолација	нема
Максимална фреквенција	50Hz (отпорен потрошувач)
	0.5Hz (индуктивен потрошувач)
PWM максимална фреквенција	0.5KHz (отпорен потрошувач)
Заштита од краток спој	да
Индикација од краток спој	Со помош на software
Напојување на излезите	
Работен напон	20.4 до 28.8VDC
Номинален напон	24VDC

#### Аналогни излези

Број на излези	2
Опсег на излез	0-10V, 4-20mA
Резолуција	12-bit (4096 единици)
Импеданса	1k $\Omega$ минимум-напон
	500 $\Omega$ минимум-струја
Галванска изолација	нема
Линеарна грешка	$\pm 0.1\%$
Работна гранична грешка	$\pm 0.2\%$

#### Дисплеј

Бид на дисплеј	STN, LCD дисплеј
Осветлување	бела LED
Резолуција на дисплејот	128x64 пиксели
Површина	2.4"
Контраст на екранот	со помош на софтвер
Тастатура	20 копчиња

## Програм

Работни видови	Количина	Симбол	Вредност
Memory Bits	4096	MB	Bit
Memory Integers	2048	MI	16-bit
Long Integers	256	ML	32-bit
Double Word	64	DW	32-bit
Memory Floats	24	MF	32-bit
Timers	192	T	32-bit
Counters	24	C	16-bit

## Одстранлива меморија

Micro SD компатибилна картичка до 2GB меморија

## Комуникациски приклучоци

Приклучок 1	1 канал, RS232/RS485
Галванска изолација	нема
Пренос на податоци	300 до 115200 bps

## RS232

Влезен напон	±20VDC
Должина на кабел	15m max (50)

## RS485

Влезен напон	-7 to +12VDC
Вид на кабел	оклопени усукани парови
Должина на кабел	1200m maximum (4000')

## Проширување на влез

Локално	Со помош на влезно/излезен приклучок за проширување, може да се направи целина до 8 проширувачки модули
Одалечени	влезови/излези. Потребен адаптер (P.N. EX-A1). Со помош на CANbus приклучок

## Други елементи

Часовник	Време и датум
Батерија	7 години на 25 <sup>0</sup> С,е подрзана од RTC и system data,
	вклучува вариабилна data.
Големина	109 x 114.1 x 68mm (4.29 x 4.49 x 2.67")
Тежина	227g
-околина	
Работна температуре	0 до 50°C (32 до 122°F)
Температура	-20 до 60°C (-4 до 140°F)
Релативна влажност	10% до 95%
Методи на монтирање	Panel монтирање (IP65/NEMA4X)
	DIN-rail монтирање (IP20/NEMA1)

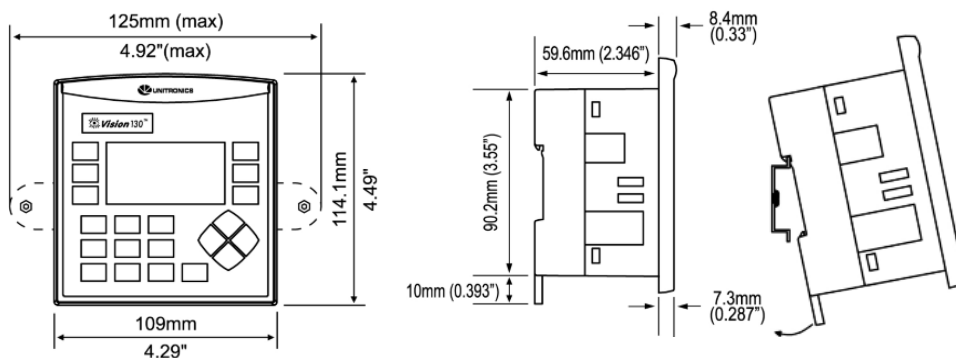
### 3.3. Инсталација

Програмибилниот логички контролер V130-33-TA24 содржи 12 дигитални влезови конфигурирани со помош на жичано поврзување во кој се вклучени 2 аналогни, 2 термоелементи/PT100 еден HSC/Schaft-encoder влезови. На излез има 10 транзисторски и 2 аналогни излези. Влезовите и излезите се конфигурирани на самата печатена плочка. Контролерот содржи и 2.4" екран и тастатура.



Слика 3.2: Поврзување на PLC со GSM модем

За комуникација имаме: приклучок RS232/RS485 и опциски Ethernet приклучок исто така и еден опциски приклучок за CANbus. Контролерот подржува SMS, GPRS, MODBUS. Протоколот FB овозможува програмибилниот логички контролер да се врзе со други уреди со помош на RS232/RS485/Ethernet приклучоците. Information Mode ни овозможува мониторинг и уредување на работните величини, регулирање на контраст и острина на екранот, стоп, формат, ресет на контролерот, се активира со притискање на тастерот <i>. CD кое кое се добива во пакет со контролерот содржи VisiLogic софтвер и комуникациски додатоци. Софтверот овозможува конфигурирање на хардверот и пишување на HMI и Ladder управувачки апликации, содржи и библиотека за на подпрограми како помош при решавање на сложени задачи.

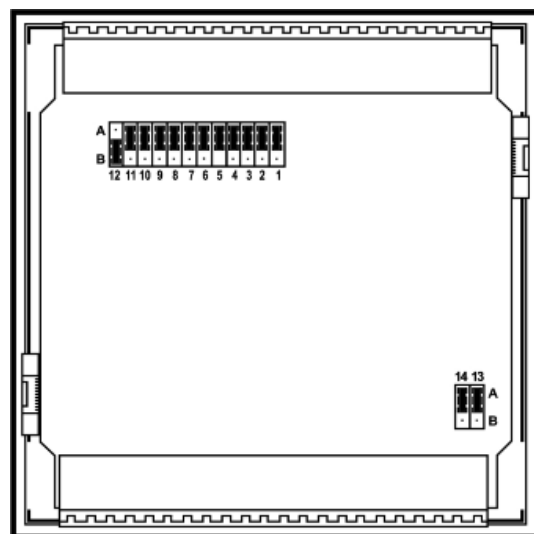


Слика 3.3: Димензии и монтирање на PLC

### 3.3.1. Конфигурација и поврзување на влезови и излези

Овој модел има вкупно 12 влезови од кои 10 дигитални и 2 аналогни влеза. Влезовите со помош на jumper можат да се одредуваат како npn или pnp. Влезовите 5 и 6 може да функционираат како дигитални или како аналогни влезови. Влезот 0 може да функционира како брз бројач, како shaft-encoder или како нормален дигитален влез. Влезот 1 може да функционира како бројач, нормален дигитален влез или како shaft-encoder. Влезовите 7-8 И 9-10 може да функционираат како дигитални или термоелементи. Влезот 11 може да послужи како CM сигнал за PT100 термоелемент. Табелите ни покажуваат како може со помош на џамперите да се уредува функционалноста на влезовите. До нив можеме да дојдеме ако се отвори контролерот.

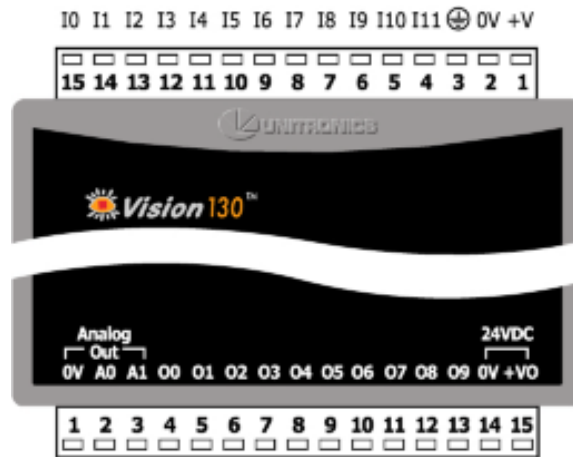
Digital Inputs 0-11: Set Type			
Set to	JP12 (all Inputs)		
nnp (sink)	A		
pnp (source)*	B		
Inputs 7/8: Set Type - Digital or RTD/TC #1			
Set to	JP1	JP2	JP3
Digital*	A	A	A
Thermocouple	B	B	B
PT100	B	A	B
Inputs 9/10: Set Type - Digital or RTD/TC #0			
Set to	JP5	JP6	JP7
Digital*	A	A	A
Thermocouple	B	B	B
PT100	B	A	B
Input 11: Set Type - Digital or CM for PT100			
Set to	JP11		
Digital*	A		
CM for PT100	B		
Input 5: Set Type - Digital or Analog #3			
Set to	JP4	JP10	
Digital*	A	A	
Voltage	B	A	
Current	B	B	
Input 6: Set Type - Digital or Analog #2			
Set to	JP8	JP9	
Digital*	A	A	
Voltage	B	A	
Current	B	B	



Analog Output 0: Set to Voltage/Current		
Set to	JP13	
Voltage*	A	
Current	B	

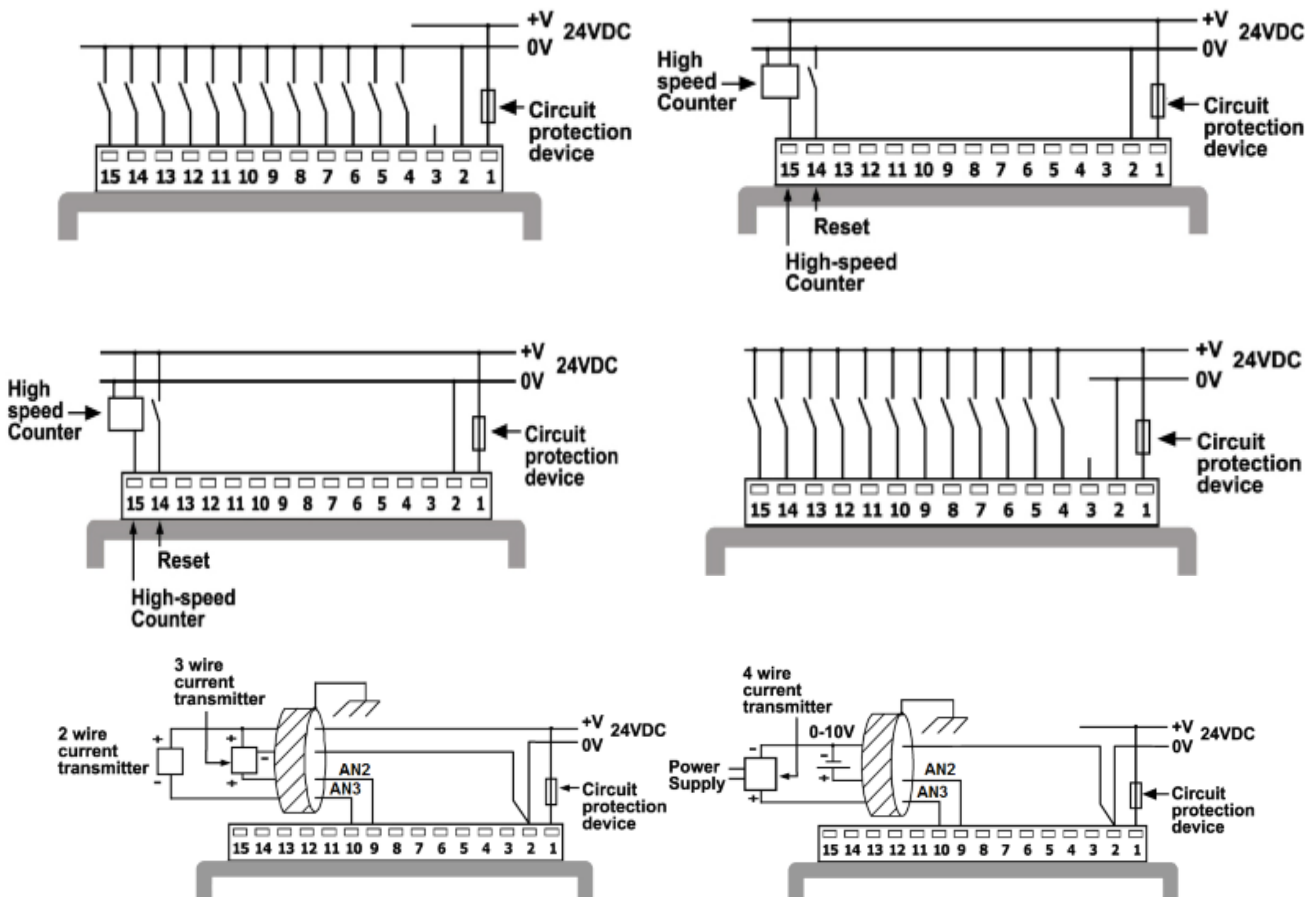
Analog Output 1: Set to Voltage/Current		
Set to	JP14	
Voltage*	A	
Current	B	

Поврзување на влезовите / излезите

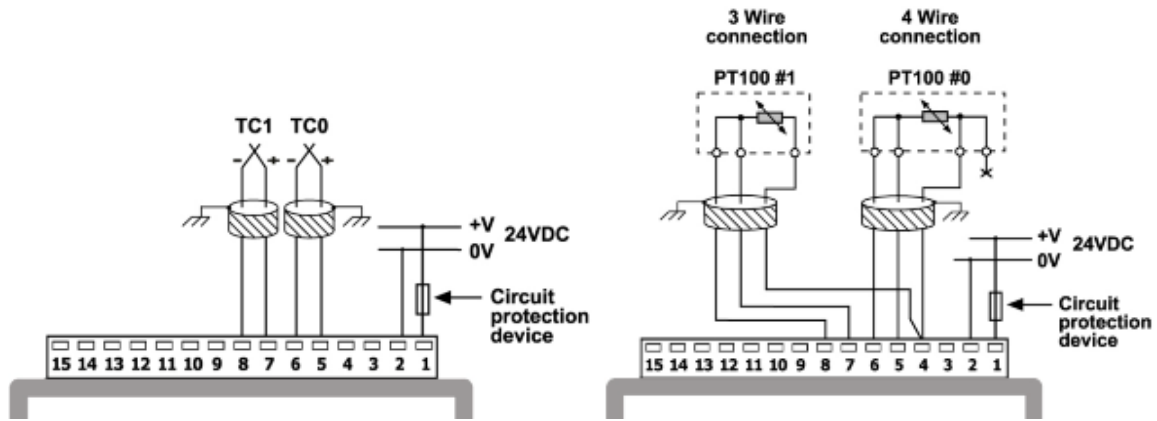


Слика 3.4: Влезно излезни клеми

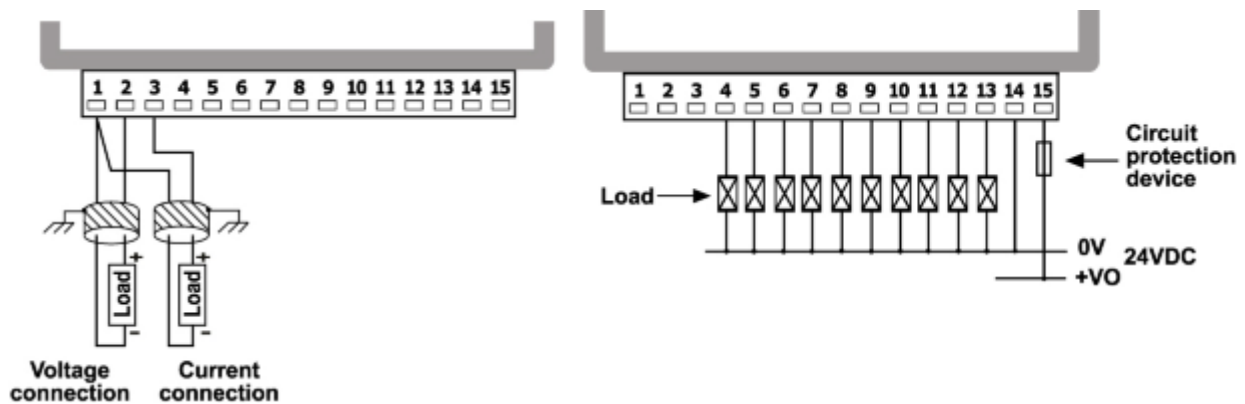
Поврзувањето на влезните аналогни сензори е прикажано на слика 3.5



Слика 3.5: Поврзување на влезните аналогни сензори



Слика 3.6: Поврзување на влезните PT100 сензори



Слика 3.7: Поврзување на излезните уреди

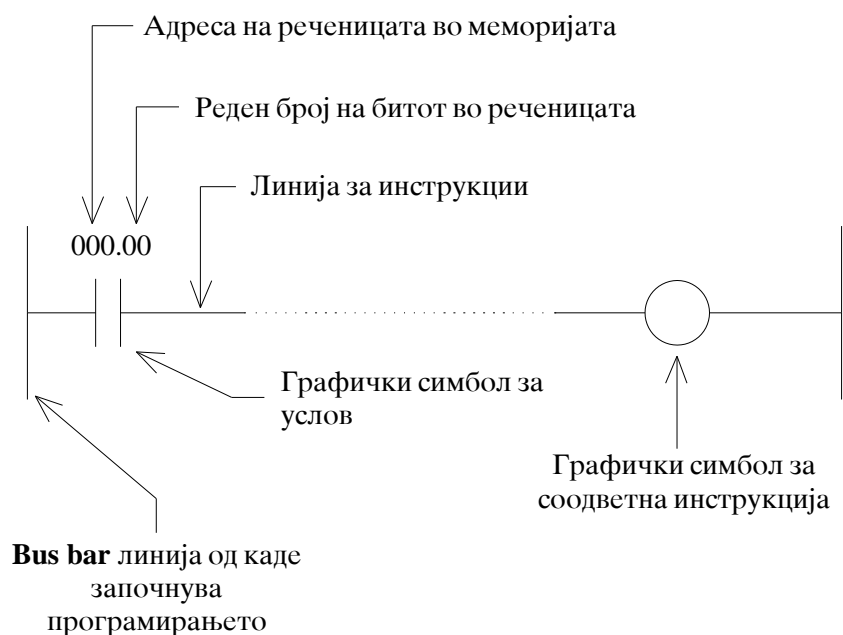
### 3.3.2. Комуникациски PORT

Овие приклучоци служат за поврзување на контролетот со други уреди. Пред да се изврши приклучување контролерот треба да се исклучи од напојување. Сигналите ја користат истата 0 од изворот на напојување. PORT 1 може да се употребува или како RS232 или RS485 со помош на jumper. Со RS232 контролерот се врзува со персонален компјутер со што се симнува соодветна програма која овозможува контролерот да се поврзе со системи како што е SCADA. Приклучокот RS485 овозможува контролерот да се врзе во мрежа со други уреди (до 32 уреди).



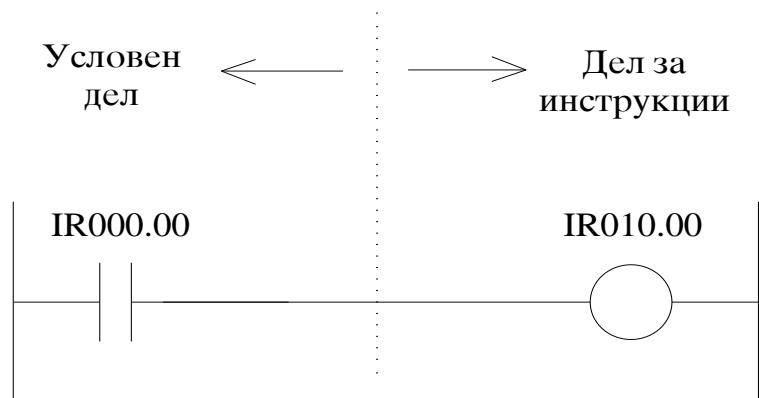
## 4. Софтверски пакет Visilogic за програмирање на PLC

Програмибилниот логички контролер V130 33-TA24 се програмира со помош на Ladder логика. Ледер (Ladder) дијаграмот претставува најпристапен начин за програмирање на PLC. Инструкциите кои ги користи овој дијаграм најлесно можат да се поделат на влезни, кои ги дефинираат условите и излезни кои се извршуваат кога условите се исполнети. Со нивна комбинација се создаваат логички блокови кои ја носат логиката на системот кој се автоматизира. Ледер дијаграмот се состои од една вертикална линија, која се нарекува **bus bar** линија. Хоризонталната линија која се граничи со **bus bar** линијата и се простира кон десно, се нарекува линија за инструкции. Вдолж оваа линија се поставуваат услови кои водат до инструкцијата сместена на десниот крај од дијаграмот. Функцијата ќе биде извршена во моментот кога логичката комбинација од услови ќе биде исполнета. Основните елементи на Ледер дијаграмот се прикажани на слика 4.1.



Слика 4.1: Основни елементи на Ледер дијаграмот

Инструкциите користат најмалку еден операнд, но најчесто и повеќе од нив. Во горниот пример операнд е 0 во мемориската локација 000.00. Како операнд се среќава мемориска локација, соодветен бит во мемориска локација или пак конкретна нумеричка вредност. Кога се користи конкретна нумеричка вредност, односно бројна константа, секогаш пред неа стои знакот #. Компајлерот кога ќе забележи знак # знае дека се работи за константа, а не за адреса. Во принцип Ледер дијаграмот се состои од два основни дела. Условниот дел, е местото од линијата за инструкции на кое се поставува логичка комбинација од услови. Во конкретниов пример, за да се задоволи условот потребно да се активира битот 00 во мемориската локација 000 Оваа адреса е директно поврзана на влезна линија од контролерот што ни дава можност за директно менување на состојбата. Секој услов во Ледер дијаграмот може да има состојба ON (логичка единица) и состојба OFF (логичка нула).




Слика 4.2: Условен и инструкциски дел

Во конкретниот случај кога битот добива логичка единица, условот е исполнет, сигналот тече понатаму кон исполнување на зададената функција. Десниот дел од Ледер дијаграмот е резервиран за логичката функција, која може да биде едноставна или сложена. На горниот пример, прикажана е едноставна функција, која се состои во активирање на битот 00 во мемориската локација 010.

## VISILOGIC Ледер Едитор

Ледер едиторот се користи за креирање на ледер дијаграм кој ја вклучува вашата управувачка апликација. Ледер дијаграмите се составени од: контакти, намотки и функциски блок елементи распределени во мрежи. Во Ледер дијаграмот, контактите ги претставуваат влезните услови. Тие се управуваат од левата кон десната Ледер линија. Затоа првиот елемент во мрежата мора секогаш да ја допира левата линија. Намотките ги претставуваат излезните инструкции. За да може излезните намотки да бидат активни, логичката состојба на контактите мора да дозволува напојувањето да тече низ мрежата кон намотката. Заради тоа елементите во мрежата мора да бидат поврзани. Секоја мрежа мора да содржи само една линија. Ледер едиторот се употребува за:

- Поставување и поврзување на ледер елементи.
- Примена – Споредба: Математички, Логички, Часовник, Зачувувачки, и Векторски функции.
- Вметнување на функциски блокови (FBs) во вашиот програм.
- Создавање на програмски модули и потпрограми, и употреба на потпрограмски скокови и ознаки.
- Коментирање на ледер мрежите.

Ледер елементите и функциите можат да се повлекуваат помеѓу мрежите. Специјалните копчиња, се исто така достапни за лесно програмирање. За стартување на ледер едиторот се кликува на копчето  на алатникот.

### Избирање на Ледер Елементи и Функции

Постојат различни начини за да се изберат елементи и функции:

- Притиснете го елементот од алатникот.
- Изберете елементи, функции и функцииски блокови од Ледер менито.
- Десен-клик на мрежата и изберете го посакуваниот елемент.

Алатник за коментари  
Додадете коментар  
на вашиот програм.

### Навигација на Проектот

- Кликнете на предметот за да го отворите.
- Десен-клик за да додадете, исчистите, избришете или промените име на Модули, Потпрограми и Екрани.

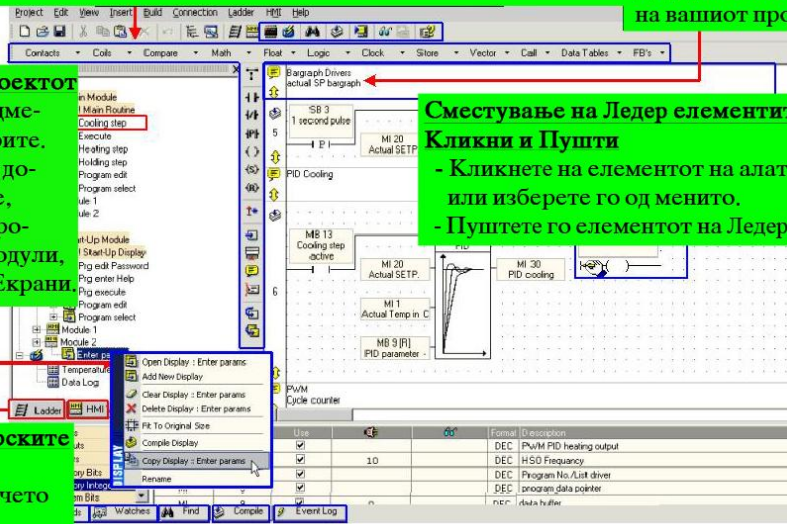
### Сместување на Ледер елементите :

#### Клики и Пушти

- Кликнете на елементот на алатникот или изберете го од менито.
- Пуштете го елементот на Ледер мрежата.

### Поглед на Тастерските Префрлувачи

- Кликнете на копчето за едитирање.



Слика 4.3: Работна околина во VISILOGIC

## Сместување на Ladder елементите во мрежата

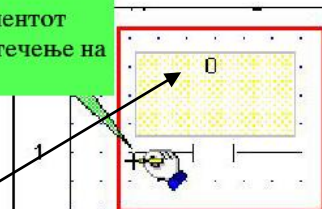
1. Се селектира Ladder елементот.

Одберете Елемент

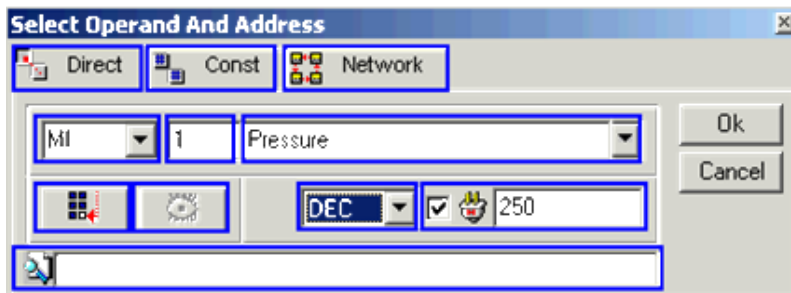


2. Се поместува на посакуваното место, со клик се појавува дијалогот за операнд.

Задвижете го елементот на мрежата.  
Забелешка : Елементот мора да допушта течење на струја



На горното место над елементот се одредува операндот и адресата.



## Бришење на Елементи

**Изберете Еден Елемент**  
Кликнете го, и елементот ќе добие сива боја.

За да го преместите елементот во друга мрежа кликнете Cut

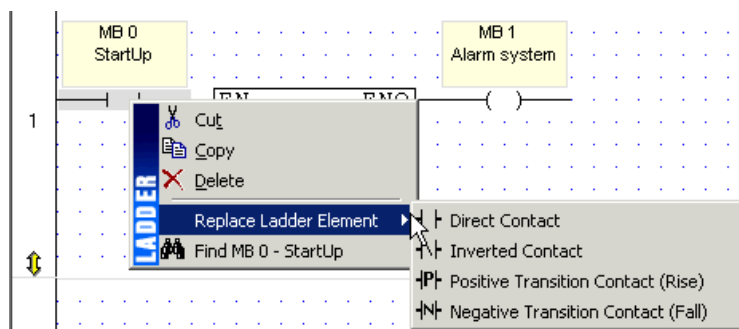
За да ги копирате селектираните елементи, кликнете Copy

**Изберете Повеќе Елементи**

1. Кликнете веднаш до групата елементи која сакате да ја селектирате
2. Задржете го левото копче од глушецот и повлечете со стрелката преку посакуваната група.
3. Кога ќе го отпуштите копчето селектираните елементи ќе добијат сива боја.

или притиснете delete на тастатурата

## Промена на типот на елементот



За да го промените типот на елементот откако е поставен на мрежата и е поврзан со операнд треба да се кликне со десен клик на елементот, изберете „Replace Ladder Element“, потоа изберете го потребниот тип на елемент.



**Нормално отворен контакт**-има функција како отворен тастер. Ако работниот bit е 1 на влез имаме 1 тогаш тогаш на излез имаме 1, а ако имаме мирен бит односно 0 тогаш контактоот не проведува.

Операнди кои се поврзуваат:

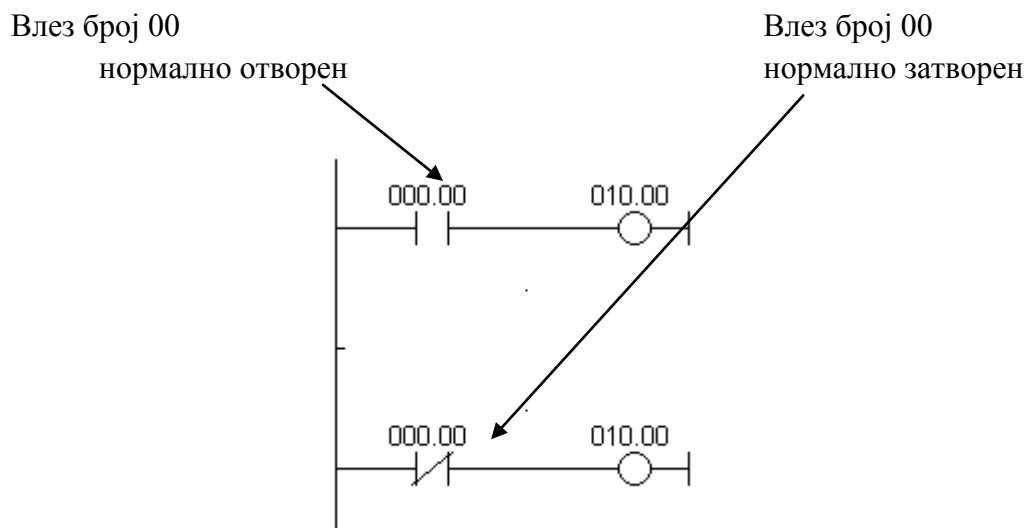
- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Output
- Timer



**Нормално затворен контакт**-има функција како затворен тастер. Во нормална состојба кога имаме на влез 1 контактоот проведува работниот bit е 0, а кога имаме работен бит 1 овој контакто е прекинат не проведува. Со овие два контакта може да се направат сите логички функции.

Операнди кои се поврзуваат:

- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Output
- Timer



**Слика 4.4:** Пример за нормално отворен и нормално затворен контакт на влез на PLC уред



Намотка која се наоѓа на излез и се активира кога условот е исполнет.

Операнди кои се поврзуваат:

- Memory Bit
- System Bit
- Output
- Timer



Намотка која се исклучува кога условот ќе се исполни.

Операнди кои се поврзуваат:

- Memory Bit
- System Bit
- Output
- Timer

## Операнди

Ледер елементите и функциите се поврзани со операнди. Операндите содржат податоци. Ледер елементите и функциите го одредуваат начинот на кој податоците се употребуваат во програмот. Секој операнд има своја адреса и опис. Кога ќе изберете Ледер елемент и ќе го поставите на мрежа, избраниот Операнд и Address box-от се отвара, овозможувајќи ви да ги поврзете типовите на операнди, да изберете адреса, и да доделите опис. Забележете дека можете да додавате вредности и забелешки на излезниот прозорец.

За да се види листата на операнди:

1. Се избира „Operand“ во долниот дел на прозорецот при што операндите ќе се прикажат.
2. Со кликување на типот на операндот во левата табела ќе се покаже листа за тој операнд.

## Типови на Операнди и Симболи

Тип	Симбол	Квантитет	Вредност	Ранг на Адреса
Влез	I	544	Bit	I0-I543
Излез	O	544	Bit	O0-O543
Тајмер	T	192	32-bit	T0-T191
Бројачи (C)	C	24	16-bit	C0-C24
Мемориски Bit	MB	4096	Bit	MB0-MB4095
Мемориски Цел Број	MI	2048	16-bit	MI0-MI2047
Мем. Цел Број (со повеќе места)	ML	256	32-bit	ML0-ML255
Двоен Слог (без предзнак)	DW	64	32-bit	DW0-DW63
Мем. Цел Бр.со подвижна запир.	MF	24	32	MF0-MF24
Константна Вредност	#	Динамички		Динамички

## Системски Операнди

Системските Операнди се поврзани со евентуалните функции и вредности во

оперативениот систем на контролерот.

Тип	Симбол	Квантитет	Вредност	Ранг на Адреса
Системски Bit	SB	512	Bit	SB0-SB511
Системски Цел Број	SI	512	16-bit	SI0-SI511
Системски Цел Бр. со повеќе места	SL	56	32-bit	SL0-SL63
Системски Двоен Слог (недоделен)	SDW	64	32-bit	

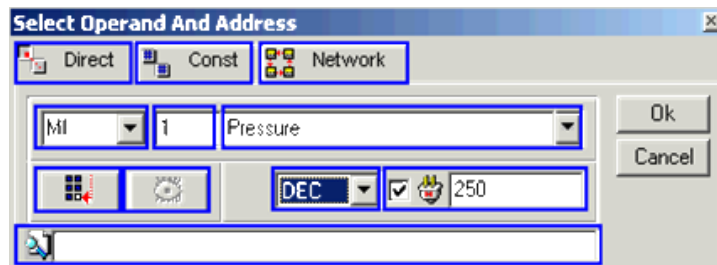
### Типови на Мрежни Операнди и Симболи

Ако контролерот е мрежно поврзан следните операнди се достапни на останатите контролери:

Тип	Симбол	Квантитет	Вредност	Ранг на Адреса
Мрежен Системски Bit	NSB	8	Bit	SB200-SB207
Мрежен Влез	NI	17	Bit	I0-I16
Мрежен Системски Цел Број	NSI	2	16-bit	SI200-SI201

### Поврзување на Операнди со Елементи

Кога ќе се поставите Ледер елементот или функцијата на мрежа, се отвара дијалогот „Select Operand and Address“. Сите операнди и типови на операнди кои се прикажани во „Select Operand and Address“ одговараат на елементите од функцијата која сте ја избрале. За да се прикаже операндот кој е придоден на елементот, се кликува двојно на насловот со жолта боја на елементот откако ќе се постави на Ледер дијаграмот. Може да се пребарува со цел да се пронајде конкретен операнд со употреба ја опцијата „Search“, во долниот дел од Dialog Box–от.



### Адресирање на Операнд

Адресата на операндот е физичка локација во меморијата на контролерот каде што се сместени податоците. На пример:

MB 10 - "10" е адресата на MB Операндот

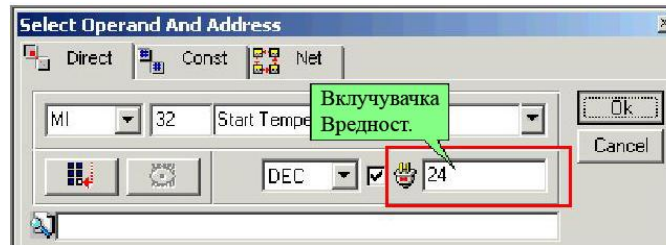
MI 35 - "35" е адресата на MI Операндот

T 12 - "12" е адресата на Тајмер Операндот

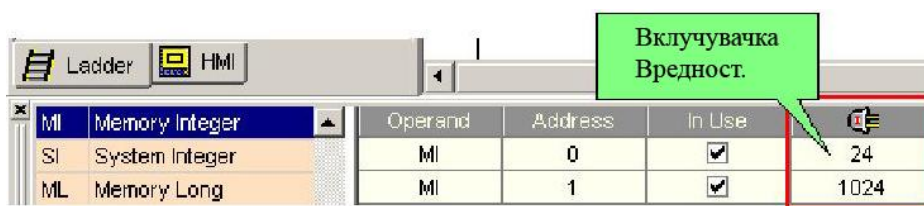
Исто така може да се додаде и објаснување за операндите кои се користат во специфична апликација.

### Вклучувачки Вредности

Вклучувачките вредности можат да бидат додадени на повеќето операнди. Вредностите се запишани во операндите при вклучување на контролерот. Бит операндите можат да бидат поставени (сетирани) или ресетираани. Вклучувачки вредности може да се доделат на операндите од тип цели броеви, цели броеви со повеќе места и двојни слогови. Активирање на вклучувачките вредности се изведува во dialog box-от за операнд и со впишување на вредност во полето обележано со црвена боја:



Алтернативен начин на внесување на вклучувачка вредност преку прозорецот во кој се прикажува информација за операндите.



## Константни Вредности #

Константна Вредност е цел број, со или без предзнак креиран од страна на програмер. Константните вредности се означени со број. За да се употреби константна вредност при програмирање се избира ја опцијата „Constant“ во „Select Operand and Address“ dialog box-от и се внесува посакуваната вредност. Исто така можете да се избере опцијата, цел број, без предзнак.



## Константни вредности на операндите

Во секоја апликација може да се создаде листа на именувани константни вредности на операндите на излезниот прозорец во долниот дел на екранот. Се избира го „Constant tab“ во излезниот прозорец при што се отвора листата со константни вредности. Се внесува опис и вредност при што треба да се обрне внимание на недоделената опција. Со внесување опис на константна вредност во „Select Operand and Address dialog box“, можете да ја користите оваа вредност во вашата апликација.

## Типови на Операнди

### Мемориски Bit-ови (MB)

Мемориските Bit-ови се bit операнди (0 или 1) и постојат 4096 Мемориски Bit-ови, со адреса MB 0 - MB 4095.

### Влезови (I)

Влезовите се bit операнди (0 или 1). Бројот на влезовите варира во зависност од Snap-in I/O Модулите и I/O проширувачките модули, кои се интегрираат во соодветната апликација. Влезот е постоечка хардверска поврзувачка врска со контролерот.



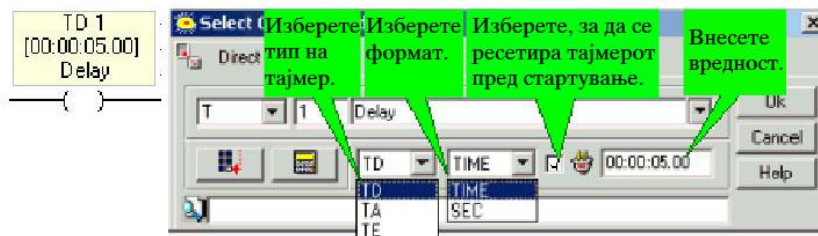
## Излези (O)

Влезовите се bit операнди (0 или 1). Бројот на излезите варира во зависност од Snap-in I/O Модулите и I/O проширувачките модули, кои се интегрираат во соодветната апликација. Излезот е постоечка хардверска поврзувачка врска со контролерот.

## Тајмери (T)

Тајмерите се елементи кој овозможуваат управување со операциите поврзани со време. Максималниот опсег на време на кој може да се програмираат тајмерите е 99 часа, 59 минути и 59.99 секунди. Има три типа на тајмери:

- TD- Timer On Delay (тајмер со доцнење при вклучување)
- TA Timer Accumulated (акумулирачки тајмер)
- TE Timer Extended Pulse (тајмер со продолжен импулс)

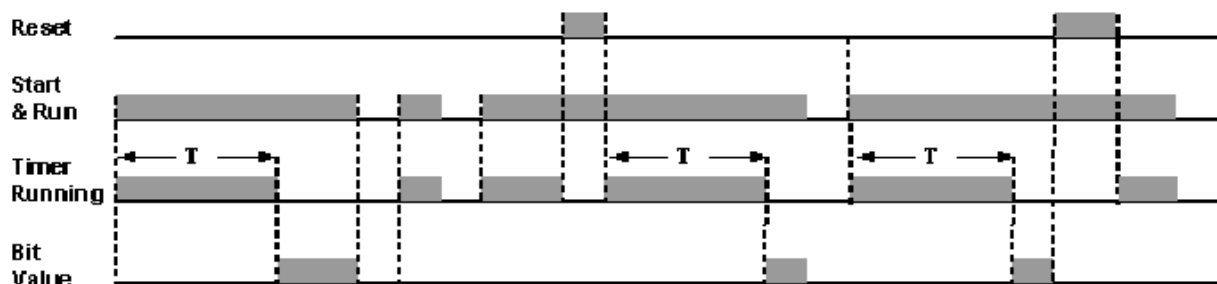


Секој тип на тајмер има три варијанти:

- Тајмерска Bit Вредност: Тајмерот се скенира како bit – тип на податок (скенирање за OFF, скенирање за ON). Резултатот од скенирањето зависи од типот на тајмерот.
- Пресетирачка вредност на тајмерот. Стартувачкиот тајмер секогаш опаѓа (се намалува) од пресетираната вредност. Пресетирачките вредности на тајмерот, за сите тајмери се вчитуваат на стартувањето. Пресетирачката вредност на тајмерот е исто така вчитана во постоечката вредност кога тајмерот се ресетира.
- Тековна вредност на тајмерот. Тековната вредност на тајмерот зависи од типот на тајмерот.

Сите типови на тајмери се активирани со покачување на преодниот рабна условниот бит, од состојба OFF во состојба ON. Условите кои се користат за да се активира тајмерот треба да се скенираат само еднаш пред скенирањето на PLC програмот.

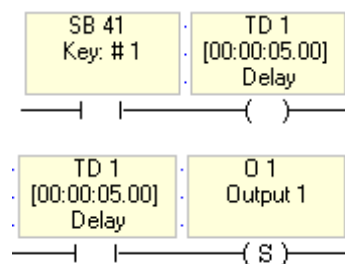
## TD- Timer On Delay



Овој тип на тајмер чека одредено време пред да вклучи. Кога на влез на тајмерот имаме 0 тогаш тајмерот не работи, на излез имаме 0.

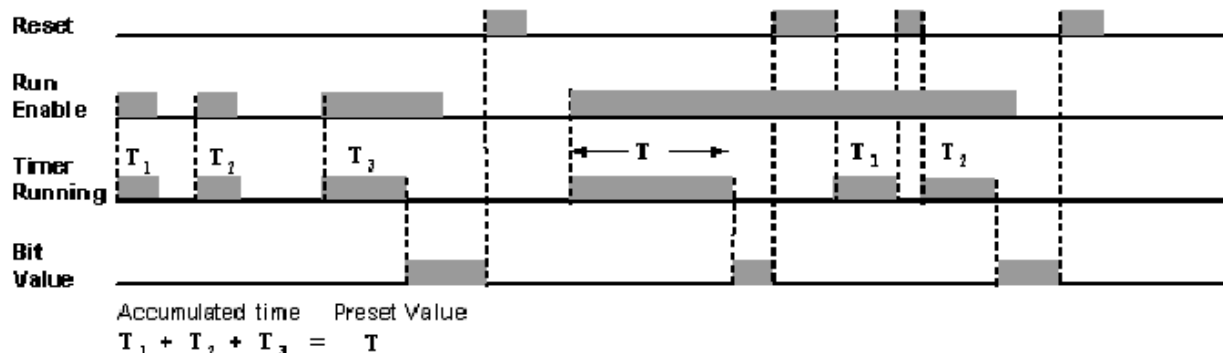
Кога на влез на тајмерот имаме 1 тогаш тајмерот отпочнува со одбројување на времето, кога ќе заврши со одбројување (до однапред зададените вредности) на излез се добива 1 се додека на влез имаме 1. Кога на влез условот е исполнет и трае пократко од preset вредноста тогаш тајмерот престанува со одбројување на времето. Тајмерот престанува со одбројување кога ќе се ресетира.

На слика 4.5 е прикажан пример за употреба на тајмер со доцнење во едноставна управувачка задача. Ако копчето број 1 на тастатурата се притисне ќе се активира тајмерот TD1, кој е пресетиран на 5 секунди. Ако копчето број 1 се задржи 5 седунди, вредноста на TD1 се намалува до нула и излезот O1 се активира, но останува активиран се додека е притиснато копчето 1. Ако копчето број 1 се отпушти пред да заврши TD1 со одбројување, тајмерот сопира. Кога копчето број 1 е притиснато повторно, TD1 повторно се враќа на 5 секунди.



Слика 4.5: Пример за примена на тајмер со доцнење кај PLC

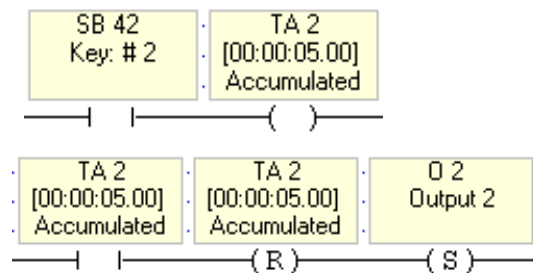
### TA Timer Accumulated



Овој тајмер овозможува собирање на времето, се додека собраното време не се изедначи со однапред зададениот опсег на време на излез имаме 0, кога ќе се исполни условот  $T_1 + T_2 + T_3 \dots T_n = T$  тогаш на излез тајмерот покажува 1. од дијаграмот се гледа дека со ресетот не се губи одброеното време. Кога preset вредноста ќе заврши тајмерот на излез ќе покаже 1 се до наредниот ресет.

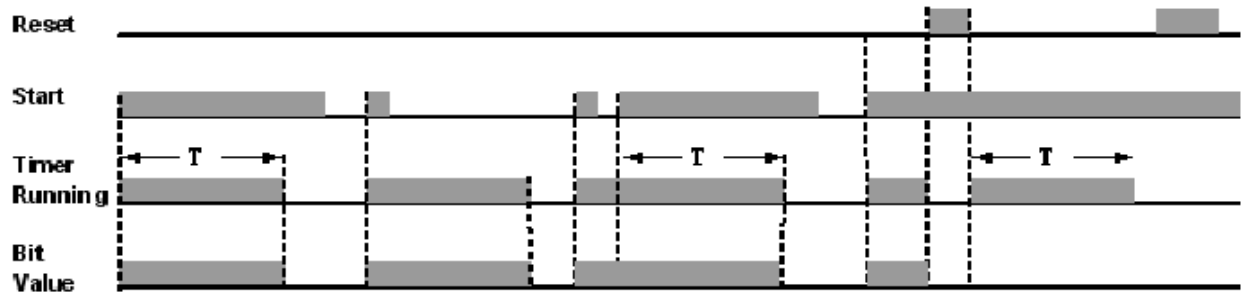
На слика 4.6 е прикажан пример за употреба на акумулирачки тајмер во едноставна управувачка задача. Ако се притисне копчето број 2 на тастатурата, се активира тајмерот TA2 кој е пресетиран на 5 секунди. Ако копчето број 2 се задржи 5 секундивредноста на тајмерот TA2 опаѓа до 0, со што се исполнува условот за вклучување на излезот O2. Излезот O2 ќе биде вклучен се до ресетирање на тајмерот TA2. Ако копчето број 2 се отпушти после 2.53 секунди, пред TA2 да ја достигне

пресетираната вредност, тајмерот запира и неговата тековна вредност е задржана . Кога Копчето број 2 ќе се притисне повторно, ТА2 продолжува да опаѓа од 2.53 секунди.



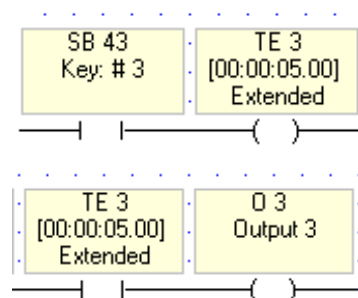
Слика 4.6: Пример за примена на акумулирачки тајмер кај PLC

### TE Timer Extended Pulse



Кога влезниот услов е исполнет односно на влез имаме 1, од тој момент тајмерот отпочнува со одбројување на времето. Со самото отпочнување на одбројувањето на времето на тајмерот на излезот покажува 1 кога ќе заврши одбројувањето на времето односно preset вредноста завршила на излезот се активира 0, односно тајмерот се исклучува. Од дијаграмот се гледа дека со краток импулс на влез тајмерот започнува со одбројување и додека трае одбројувањето на излез имаме 1. Кога ќе се изврши ресетирање тогаш тајмерот започнува со одбројување ама на излез дава 0.

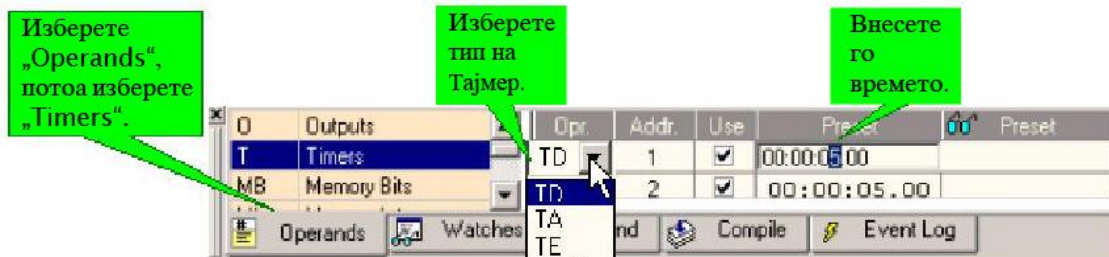
На слика 4.7 е прикажан пример за употреба на тајмер со продолжен импулс во едноставна управувачка задача. Со притискање на копчето број 3 на тастатурата се активира тајмерот ТЕ3, кој е пресетиран на 5 секунди. Откако копчето број 3 е притиснато, ТЕ3 одбројува до нула и излезот О3 се вклучува во моментот кога се вклучува тајмерот и е вклучен се додека не одброи тајмерот.



Слика 4.7: Пример за примена на тајмер со продолжен импулс кај PLC

## Подесување на тајмерите

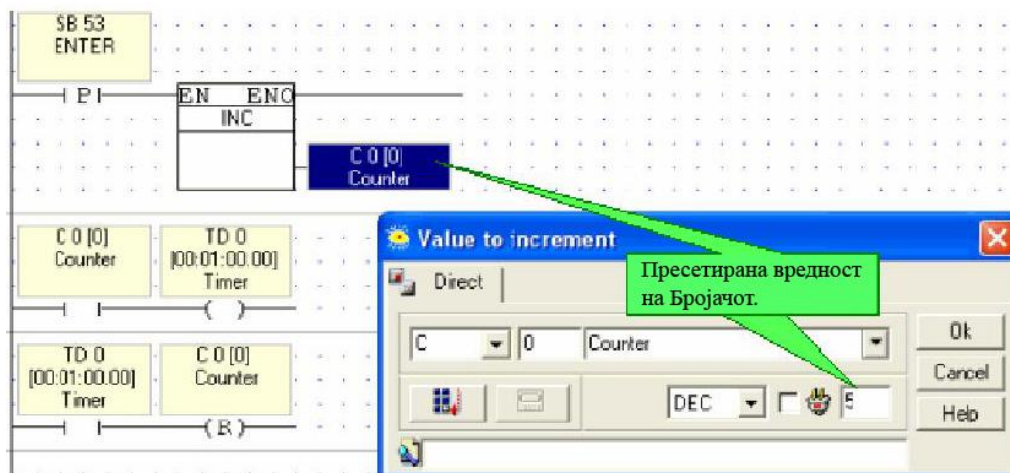
Листата на тајмери се активира со клик на „Operand“ на излезниот прозорец во долниот дел на екранот, и се избира ставката „Timers“.



Тајмерите можат да бидат сетирани и едитирани во „Select Operand and Address dialog box“ каде што се вметнува тајмерот во соодветната програма. Исто така може да се употреби и „Information Mode“ за да се промени или внесе вредност на тајмерот преку тастатурата од PLC-то додека контролерот го стартува неговиот управувачки програм.

## Бројачи ( C )

Бројачите се елементи кои овозможуваат управување со операции поврзани со одбројување. Постојат бројачи кој можат да бројат нагоре - нагорни бројачи (1, 2, 3,...), и бројачи кој можат да бројат надолу - надолни бројачи (...3, 2, 1). VisiLogic ви нуди 24 вградени бројачи, претставени со симболот C. За да се употреби „Up“ бројачот во програмата, се поставува растечка функција на мрежата и се селектира C од паѓачкото мени. За да се употреби „Down“ бројачот во програмата, се употребува опаѓачката функција. Бројачот реагира на растечки импулс кога одбројува. Кога акумулираниот број на импулси се изедначи со пресетираната вредност на бројачот, сигналот протекува низ функцијата и bit-от на бројачот се вклучува. Откако еднаш пресетираната вредност е постигната, bit-от на бројачот останува вклучен се додека не се ресетира со помош на ресетирачка намотка.



Кај нагорните бројачи - секогаш кога влезот на бројачот премине од 0 во 1 акумулаторот се зголемува за 1. Бројето започнува од вредност 0 и може да оди до

вредност 32767. Кога вредноста на бројачот ќе ја достигне однапред зададената вредност престанува со работа.

Кај надолните бројачи - секогаш кога влезот на бројачот премине од 0 во 1 акумулаторот се намалува за 1. Броњето започнува од вредност 0 и може да оди до вредност -32767. Кога вредноста на бројачот ќе ја достигне однапред зададената вредност престанува со работа.

## Преглед и избор на бројачи

Пресетираната вредност на бројачот може да биде доделена или во „Select Operand“ или во излезниот прозорец. За да се прикаже листата со бројачи треба да се кликне на „Operand“ на излезниот прозорец на дното од екранот и потоа да се избере „Counters“.



## Мемориски Цели Броеви (MI)

Мемориските Цели Броеви се 16-битни операнди кои можат да бидат означени (signed) или неозначени (unsigned). Опсегот на MI е од -32768 до +32767. Постојат 2048 MI (Адреса MI 0 - MI 2047). За да се прикаже листата на операнди, се кликува на „Operand“ на излезниот прозорец во долниот дел на екранот и потоа се избира тип на операнд.

## Мемориски Цели Броеви (со повеќе места) (ML)

Мемориските Цели Броеви (со повеќе места) се 32-битни цели броеви кои можат да бидат означени или неозначени, во низа од - 2,147,483,648 до +2,147,483,647. Постојат 256 ML (ML0 – ML255). За да се прикаже листата на операнди се кликува на „Operand“ на излезниот прозорец во долниот дел на екранот и потоа се избира тип на операнд.

## Двоен Слог (DW)

Двојни слогови (Double Words) се 32-битни неозначени целобројни операнди со максимална вредност 4,294,967,296. Постојат 64 двојни слогови, адреса DW0 до DW63.

## Мемориски Цел Број (со Подвижна Запирка) (MF)

Цели броеви со подвижна запирка се 32-битни целобројни операнди кои можат да бидат означени или неозначени, во опсег од -3.402823E38 до -1.401298E-45 за негативни броеви, и +1.401298E-45 до +3.402823E38 за позитивни броеви. Постојат 24 MF (MF 0 - MF23). За да се прикаже листата на операнди се кликува на „Operand“ на излезниот прозорец во долниот дел на екранот и потоа се избира тип на операнд.

## Системски Операнди (SI) (SL) (SB) (SDW)

Системските Операнди вклучуваат: Системски битови (SB), системски цели броеви (SI), системски двоен слог (SDW) и системски долг слог (SL). Системските операнди се употребуваат од оперативниот систем на контролерите за извршување на одредени функции и вредности. Многу системски операнди се поврзани со фиксни параметри и се „read-only“, како SB 2 стартувачкиот bit, кој се вклучува за еден циклус секогаш кога контролерот се стартува.

Останатите системски операнди можат да бидат запишани од страна на програмот, или преку „INFO Mode“. На пример, за да се пресмета моменталната внатрешна температура на контролерот, можете да го вклучите SB 14; тогаш контролерот ќе ја запише моменталната температура во SI 14, која е „read only“.

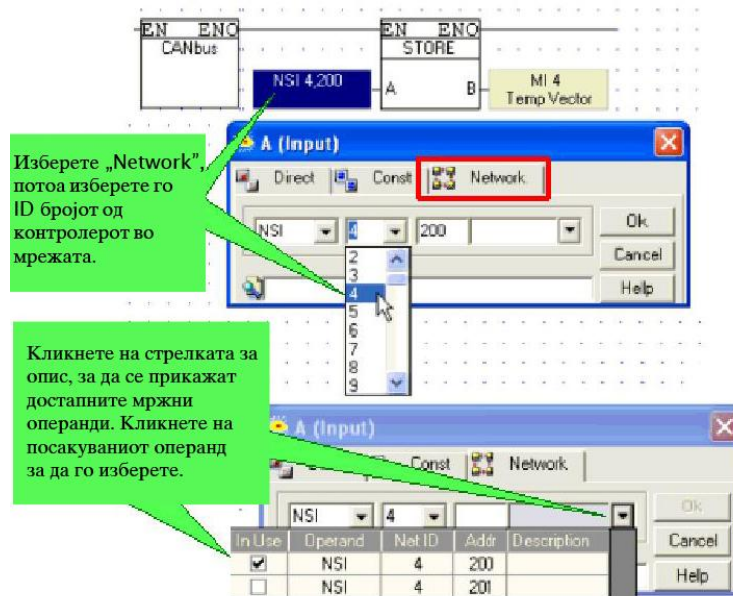
За да се прикаже листа на системски операнди со нивните описи се кликува на „Operand“ на излезниот прозорец во долниот дел од екранот и потоа се избира типот на операндот. Системските Операнди имаат однапред поставени описи кои ја објаснуваат нивната функција. Ако описите се променат, или ако отворите проект кој бил работен во друга верзија на VisiLogic, можете да го прикажете обновениот опис со помош на „Project Menu Project>System Descriptions>Restore all System Descriptions“.

Сите „SB“ и „SI“ кои немаат опис се одредени за употреба од страна на системот.

## Мрежни операнди - комуникациски податоци преку „CANbus“

Кога контролерот е интегриран во „CANbus“ мрежата, податоците содржани во поединечните системски операнди континуирано се прикажуваат на мрежата, заедно со ID (идентификациските) броеви на контролерите. Податоците се содржани во 16 системски битови (SB 200 до SB 215), 16 влеза (I 0 to I 15) и два системски цели броеви (SI 200 and SI 201).

За да се овозможи мрежниот контролер да ги чита вредностите од друг мрежен контролер се поставува посакуваната функција на мрежа. Во „Select Operand Address box“, се кликува на „Network“, потоа се избира ID од посакуваниот контролер и посакуваниот операнд.



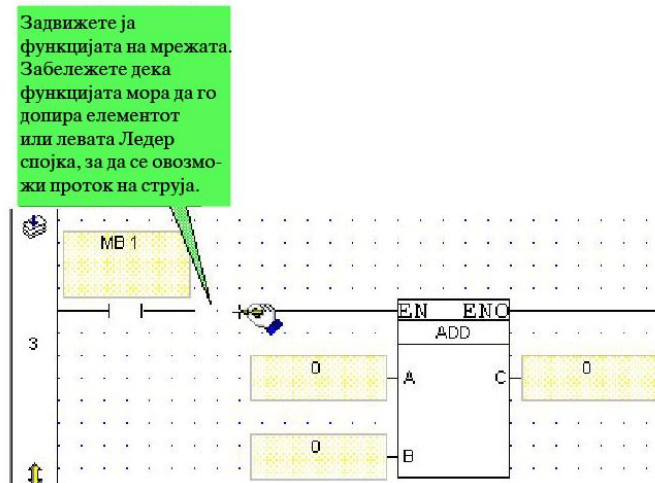
## Функции

### Поставување Функција на Мрежа

Изборот на било кој тип на Ледер функција се извршува со:

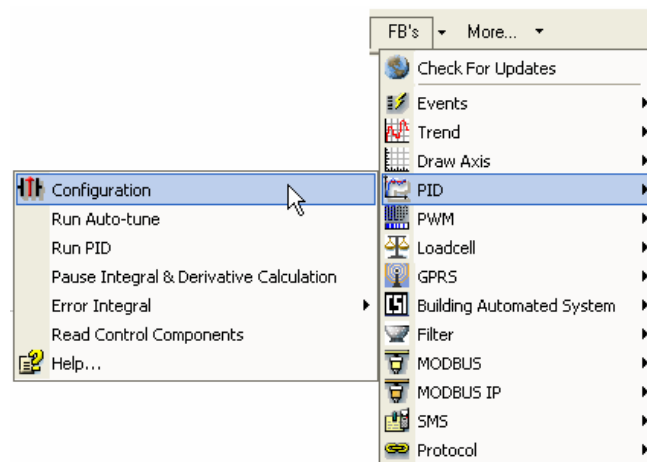
- Избирање од алатникот,
- Избирање го од менито
- Десен клик на дијаграмот и од менито се избира функција.

Функцијата се повлекува до посакуваната локација и се поврзува со операндите со употреба на „Select Operand and Address“.



### Библиотека на Функциски Блокови

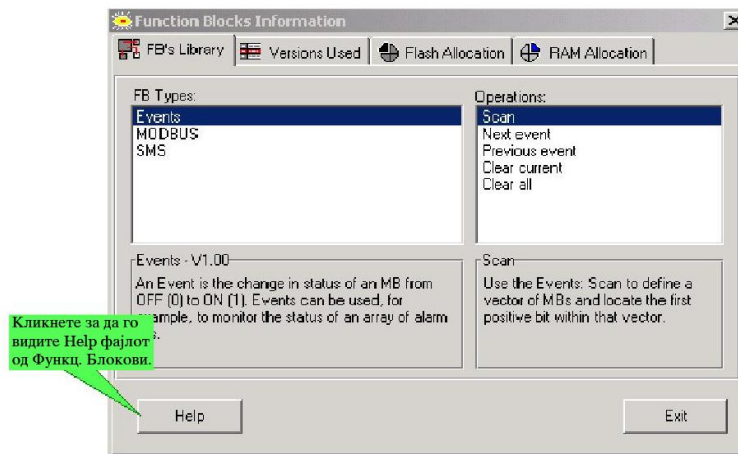
Unitronics нуди библиотека на функциски блокови за напредни функции како SMS пораки и „MODBUS“ комуникација. Функциските блокови кои се инсталирани во VisiLogic можат да се прелистаат во „FB's“ менито.



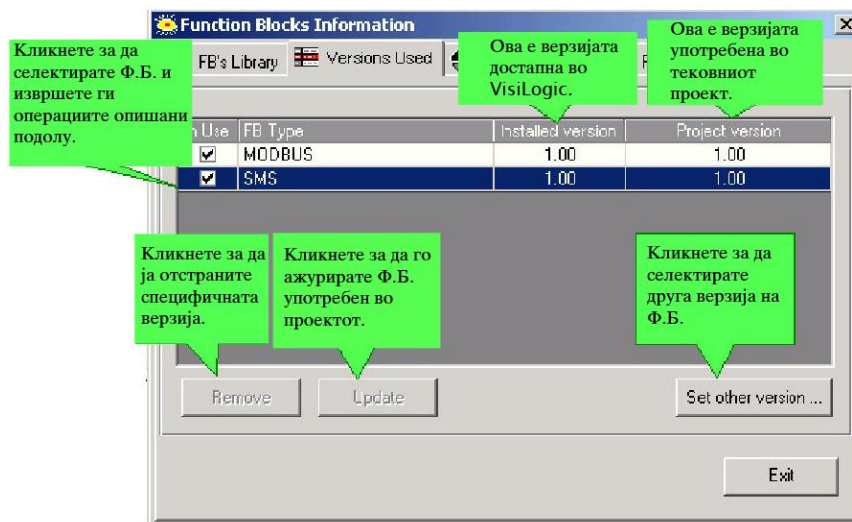
За да се инсталира ажурирана библиотека на функциски блокови се избира „Update“ од Web менито на функциски блокови или од „Help“ менито. На крајот на download-ирањето, треба да се затвори и рестартира VisiLogic. По тоа новите функциски блокови ќе се појават во менито. За да се овозможи функцијата „Live Update“, може да се употреби „proxy server“ во „Project Properties“. Со употреба на „Function Block Information“, од „View“ менито, може да се провери:

- Кои функционални блокови се инсталирани во библиотеката на Visilogic.
- Кои верзии на функционални блокови се инсталирани и кои верзии се употребени во отворениот проект, за да се управува со истите.
- Употреба на меморијата на Функционските Блокови.

Библиотеката на функционални блокови може да се види во прозорецот “Function Block Information”



Вториот таб ја покажува употребена верзија.



Во библиотеката на функционални блокови постојат повеќе типови на функции од кои можеме да ги издвоиме: Трендови: Real-Time HMI график, цртање на оска, PID функционални блок и самоподесување, MB - PWM „Loadcell“ филтер, MODBUS, serial MODBUS, IP SMS пораки, GPRS протокол за комуникација и други.






### Споредување на Функции

Споредувањето на функции споредува две вредности во зависност од типот на функцијата која сте ја избрале. Ако споредбата е вистинита (логичка 1): постои сигнал на излез од блокот.

Ако споредбата е не вистинита (логичка 0): не постои сигнал на излез од блокот.



Постојат 6 типа на споредбени функции:

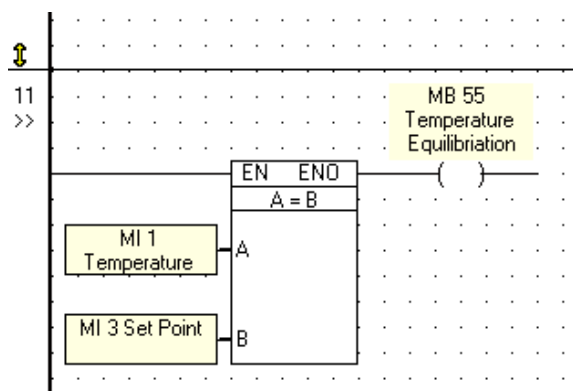
- Поголем од 
- Поголем или Еднаков 
- Еднаков 
- Не Еднаков 
- Помал или Еднаков 

Векторското мени вклучува споредбени векторски функции. Следните типови на вредности можат да се споредат.

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

### Еднаков

Функциониот блок „Еднаков“ ја споредува вредноста на влезот А со влезот В. Ако влезот А е еднаков со влезот В : ќе се појави сигнал на излезот од функциониот блок. Ако влезот А не е еднаков со влезот В: нема да се појави сигнал на излезот од функциониот блок.



Според горниот пример:

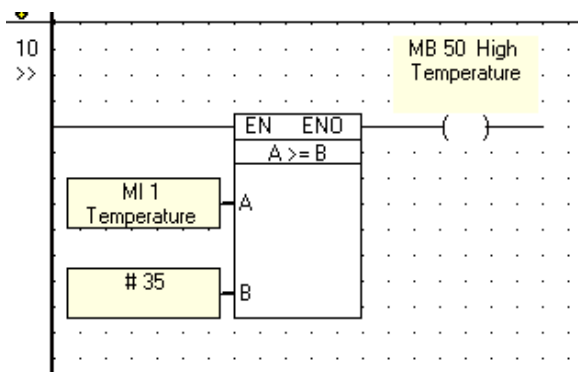
Ако MI 1 е еднаков со MI 3; тогаш MB 55 ќе премине во логичка "1" (ON). Ако MI 1 не е еднаков со MI 3; тогаш MB 55 ќе премине во логичка "0" (OFF).

Следните типови на вредности можат да се споредат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

## Поголем или Еднаков

Функцискиот Блок „Поголем или Еднаков“ ја споредува вредноста на влезот А со излезот В. Ако влезот А е поголем или еднаков со излезот В: ќе се појави сигнал на излезот од функцискиот блок. Ако влезот А не е еднаков или поголем од влезот В: нема да се појави сигнал на излезот од функцискиот блок.



Според горниот пример:

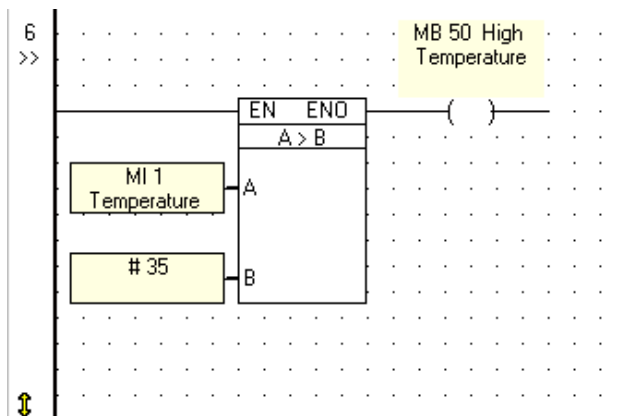
Ако MI 1 е еднаков или поголем од 35 тогаш MB 50 ќе премине во логичка "1" (ON). Ако MI 1 е помал од 35 тогаш MB 50 ќе премине во логичка "0" (OFF).

Следните типови на вредности можат да се споредат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

## Поголем од

Функцискиот Блок „Поголем од“ ја споредува вредноста на влезот А со излезот В. Ако влезот А е поголем од излезот В: ќе се појави сигнал на излезот од функцискиот блок. Ако влезот А не е поголем од влезот В: нема да се појави сигнал на излезот од функцискиот блок.



Според горниот пример:

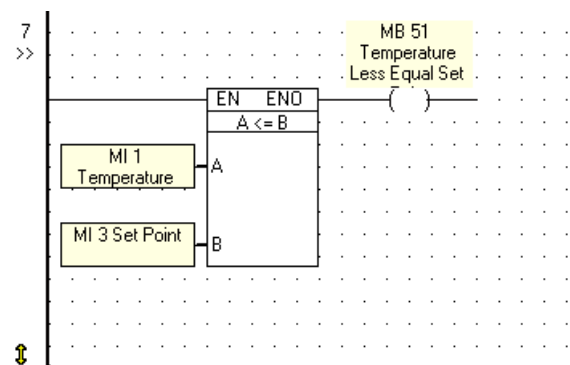
Ако MI 1 вредноста е поголема од 35 тогаш MB 50 ќе премине во логичка "1" (ON). Ако MI 1 не е поголема од 35, MB 50 ќе премине во логичка "0".  
Функциските блокови поголем и помал не даваат излез кога влезот А е еднаков со влезот В.

Следните типови на вредности можат да се споредат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

### Помал или еднаков

Функцискиот Блок „Помал или еднаков“ ја споредува вредноста на влезот А со излезот В. Ако влезот А е помал или еднаков од излезот В: ќе се појави сигнал на излезот од функцискиот блок. Ако влезот А не е помал или еднаков од влезот В: нема да се појави сигнал на излезот од функцискиот блок.



Според горниот пример:

Ако MI1 вредноста е помала или еднаква од MI3 тогаш MB 51 ќе премине во логичка "1" (ON). Ако MI1 вредноста не е помала или еднаква од MI3 тогаш MB 51 ќе премине во логичка "0" (OFF).

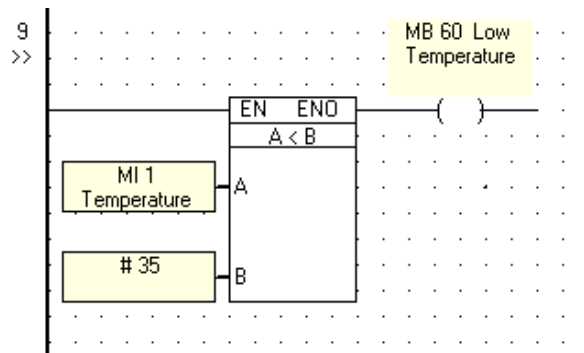
Следните типови на вредности можат да се споредат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

### Помал

Функцискиот Блок „Помал“ ја споредува вредноста на влезот А со излезот В. Ако влезот А е помал од влезот В: ќе се појави сигнал на излезот од функцискиот блок. Ако

влезот А не е помал од влезот В: нема да се појави сигнал на излезот од функцискиот блок.



Според горниот пример:

Ако MI 1 вредноста е помала од константниот цел број 35 тогаш MB 60 ќе премине во логичка "1" (ON). Ако MI 1 е поголема од константниот цел број 35 MB 50 ќе премине во логичка "0" (OFF).

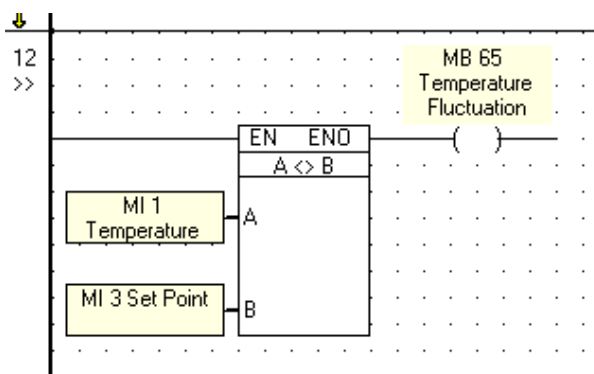
Следните типови на вредности можат да се споредат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

### Не Еднаков

Функцијата „Не Еднаков“ го проверува влезот А за да види дали излезната вредност е не еднаква со влезот В. Функцијата се наоѓа во „Compare“ менито.

Ако влезот А не е еднаков со влезот В: ќе се појави сигнал на излезот од функцискиот блок. Ако влезот А е еднаков од влезот В: нема да се појави сигнал на излезот од функцискиот блок.



Според горниот пример:

Ако MI 1 не е еднаков со MI 3; тогаш MB 65 ќе премине во логичка "1" (ON). Ако MI 1 е еднаков со MI 3; тогаш MB 55 ќе премине во логичка "0" (OFF).

Следните типови на вредности можат да се споредат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

## Логички Функции

Логичките функцииски блокови овозможуваат:

- Bit Тест
- Сетирање/Ресетирање на Bit
- И (AND)
- ИЛИ (OR)
- Нили (XOR)
- Смена (Shift)
- Ротација
- Конвертирање
- Тест Bit
- Зачувување на Bit Статус (Store Bit Status)
- Вчитување Bit Статус (Load Bit Status)
- RS-SR Флип-Флоп

Влезни вредности на логичката функција можат да бидат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Освен константната вредност, било кој од овие операнди може да ја содржи излезната вредност. Функциите се наоѓаат во „Logic“ мениот на ледер алатникот.

## И (AND)

Логичката функција И (AND) ја одредува состојбата на два цели броја. Ако bit-от е точен (логичка 1) на двата влиза А и В, тогаш излезот С ќе биде точен (логичка 1). Ако влезовите А и В се неточни (логичка 0), тогаш излезот С ќе биде неточен (логичка 0). Ако или влезот А или влезот В се неточни (логичка 0) - излезот С ќе биде неточен (логичка 0). Тоа е покажано во табелата на вистинитоста

AND Truth Table		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Влезни вредности во функцијата И (AND) можат да бидат:

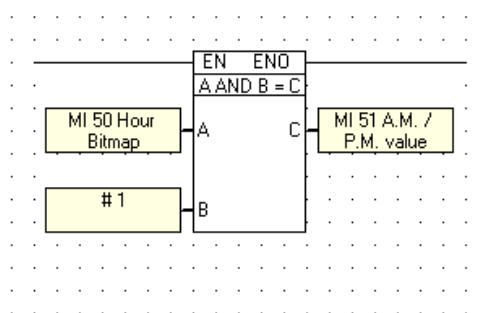
- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на Константната Вредност, било кој од овие операнди можат да содржат излезна вредност. И (AND) може да се употреби за да прикрие одредени битови на влезниот цел број кои не се релевантни за дадената функција.

**Пример:** Ако функцискиот блок часовник (clock) го употреби првиот бит од 16-битен збор за да се одреди дали даденото време е “А.М“ или “ Р.М“, можете да ги прикриете останатите 15 бита. Така ќе одредите дали тековното време е “А.М“ или “Р.М“

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1
AND																
Mask	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Result	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Сите непотребни битови ќе се исклучат (логичка 0) освен “А.М/Р.М“ битот.



Функцијата се наоѓа во „Logic“ менито, од Ледер алатникот.

## Или (OR)

Логичкиот функциски блок или може да ја одреди состојбата на два цели броја за да провери дали или А или В се точни. Ако еден од влезите А или В е точен – излезот С ќе биде точен (логичка 1). Ако двата влиза А и В се точни (логичка 1) – излезот С исто така ќе биде точен (логичка 1). Тоа е покажано во табелата на вистинитоста

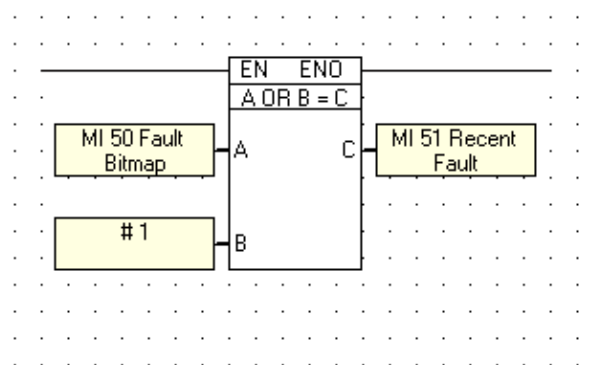
OR Truth Table		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Влезни вредности во функцијата ИЛИ (OR) можат да бидат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на константната вредност било кој од овие операнди можат да содржат излезна вредност.

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1
OR																
Compare	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Result	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1



Функцијата се наоѓаат во „Logic“ мениот на Ледер алатникот.

## Нили (XOR)

Логичката Функција Нили може да ја одреди состојбата на два цели броја за да провери дали A и B се точни. Ако влезот A или B е точен - излезот C ќе биде точен (логичка 1). Ако и двата влеза A и B се точни (логичка 1) - излезот C ќе биде не точен (логичка 0). Ако и двата влеза A и B се неточни (логичка 0) - излезот C ќе биде неточен (логичка 0).

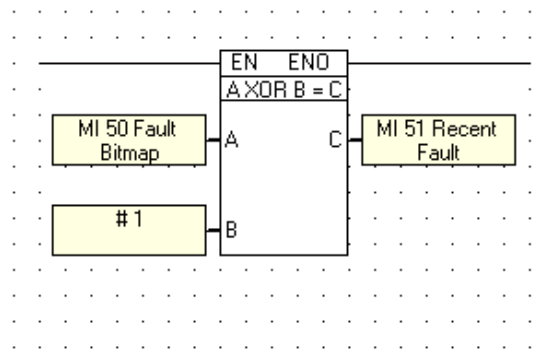
XOR Truth Table		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Влезни вредности во Нили функцијата можат да бидат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност. Употребете нили за да бидат препознаени промените во целиот број и да проверите дали има оштетен бит. Ако два цели Броја се еднакви: резултатот ќе се врати на логичка 0. Ако постои оштетен бит: оштетениот бит ќе се врати во логичка 1.

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1
XOR																
Compare	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Result	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	0

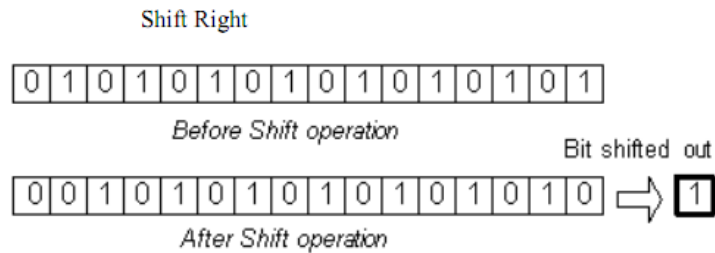


Функцијата се наоѓаат во „Logic“ менито на Ледер алатникот.

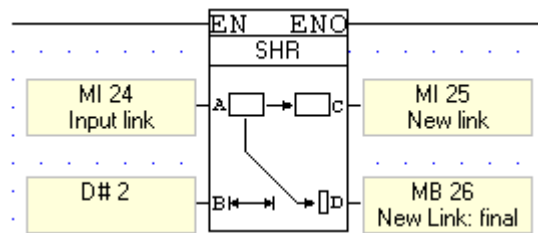
### Промена (Shift)

Функцијата „Shift“ ги задвижува битовите на целиот број на лево или на десно. Запомнете дека секој променет бит не може да се врати назад.





Операнд А: ја содржи вредноста која треба да се промени.  
 Операнд В: го содржи бројот на битови кои треба да се променат.  
 Операнд С: го содржи резултатот.  
 Операнд D: го покажува статусот на крајниот бит во целиот број после операцијата, без разлика на бројот на променетите битови за време на операцијата.



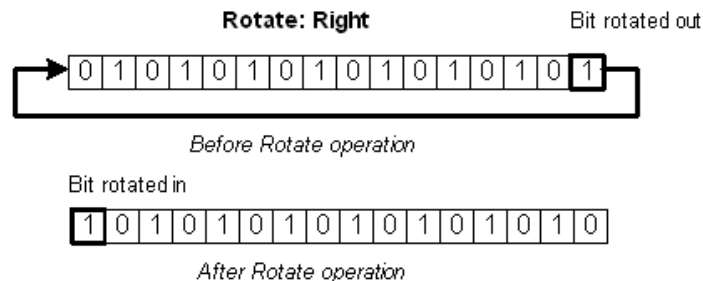
Се забележува дека без разлика на бројот на променетите битови, Операндот D го покажува статусот. Функцијата „Shift“ може да се изврши на вредности кои ги содржат следните операнди:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)

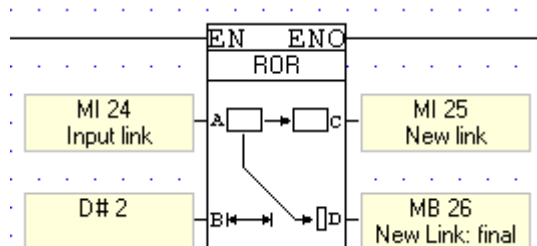
Функцијата се наоѓаат во “Logic“ менито на Ледер алатникот.

### Ротирање (Rotate)

Функцијата „Rotate“ ги задвижува битовите во целиот број на лево или на десно.



Операнд А: ја содржи вредноста која треба да се ротира.  
 Операнд В: го содржи бројот на битови кои треба да се ротираат.  
 Операнд С: го содржи резултатот.  
 Операнд D: го покажува статусот на крајниот бит после операцијата.



Функцијата „Rotate“ може да се изврши на вредности кои ги содржат следните операнди:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)

Функцијата се наоѓа во „Logic“ менито на Ледер алатникот.

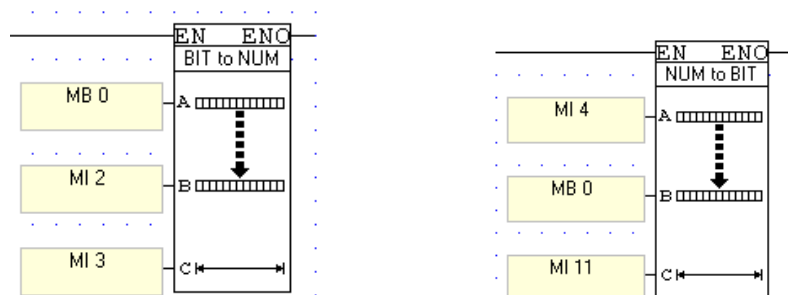
### Вектор: Bit во Нумерички, Нумерички во Bit

Со употреба на функциите може да се конвертира редот од бит вредности во нумерички вредности, или од нумерички во бит вредности. Функциите се наоѓаат во менито „Vector“.

#### Bit во Нумерички

Операнд A: ја содржи стартната адреса на редот од битови кој треба да се конвертира.  
 Операнд B: е стартот на векторот, кој ќе ја содржи конвертираната вредност. Треба да се внимава при адресирањето на операндите, ако конвертираната вредност не може да се подеси во единечниот регистар тогаш функцијата непрекинато ќе ја повторува постапката додека не ја конвертира вредноста.

Операнд C: ја содржи должината на редот кој ќе се конвертира.



#### Нумерички во Bit

Операнд A: ја содржи адресата на вредноста која треба да се конвертира.

Операнд B: ја содржи почетната адреса на редот кој ја содржи конвертираната вредност.

Операнд C: ја содржи должината на редот кој ќе ја содржи конвертираната вредност.

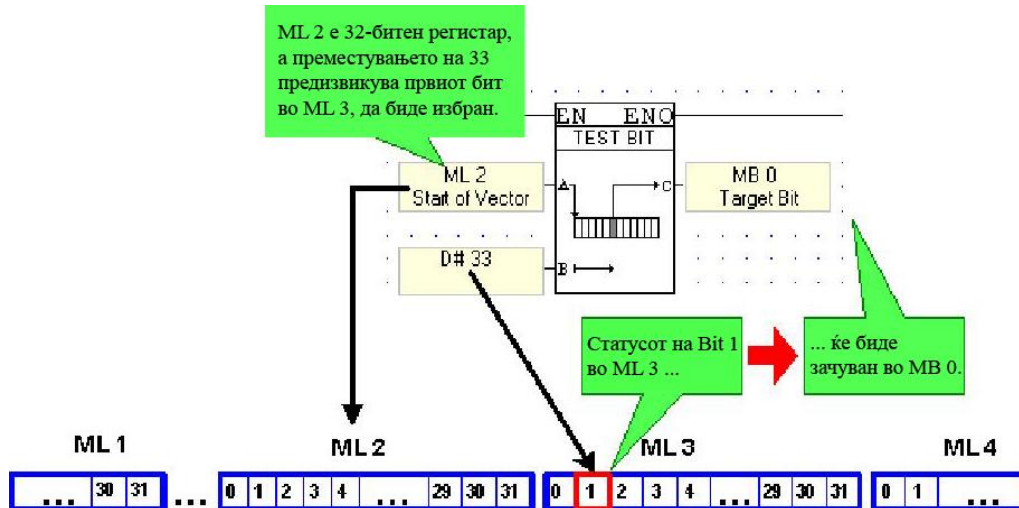
## Тестирање на Bit

Тестирање на Bit-ови овозможува да изберете bit внатре во векторскиот регистар, и да го зачувате неговиот статус во MB.

Операнд А: почеток на вектор, го одредува стартот на векторскиот регистар.

Операнд В: поместување во векторот, го избира битот внатре во векторот.

Операнд С: цел на битот, одредува каде вредноста на селектираниот бит, ќе биде сместена.



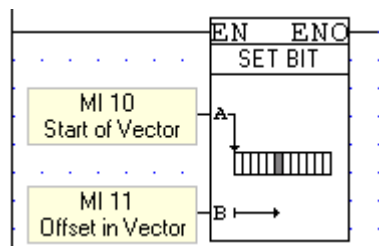
Функцијата се наоѓа во „Logic“ менито на Ледер алатникот.

## Сетирање/Ресетирање на Bit

Сетирање на Bit овозможува да се избере Bit внатре во векторот на регистрите и да се постави. Ресетирање на Bit овозможува да се избере Bit внатре во векторот на регистрите, и да се ресетира.

Операнд А: старт на вектор, го одредува стартот на векторскиот регистар.

Операнд В: поместување во векторот, го избира битот внатре во векторот.



Функцијата се наоѓа во „Logic“ менито на Ледер алатникот.

## Зачувување на „Bit Status“

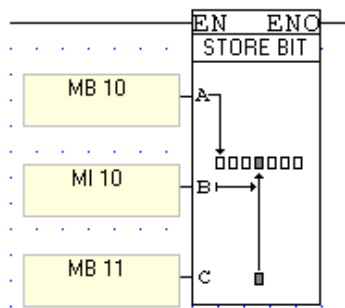
Се употребува за избор на MB и зачувување на неговиот статус во MB внатре во дефинираниот вектор.

Операнд А: старт на вектор, одредува каде векторот започнува.

Операнд В: поместување во векторот, го селектира посакуваниот bit внатре во векторот.

Операнд С: вредноста на bit-от, го одредува изворниот bit. Статусот на овој bit ќе биде

зачуван во посакуваниот bit внатре во дефинираниот вектор.



Функцијата се наоѓа во „Logic“ менито на Ледер алатникот.

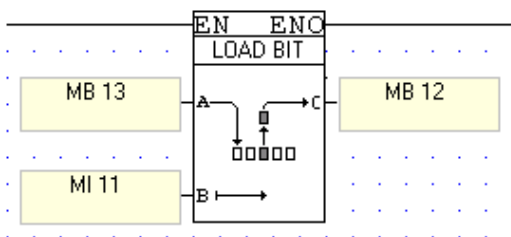
### Вчитување на „Bit Status“

Се употребува за избор на MB внатре во дефинираниот вектор и вчитување на неговиот статус во MB надвор од тој вектор.

Операнд А: старт на вектор, одредува каде векторот започнува.

Операнд В: поместување во векторот, го селектира посакуваниот bit внатре во векторот.

Операнд С: вредност на bit-от, го одредува посакуваниот bit кадешто вредноста на извориот bit ќе се зачува.



Функцијата се наоѓа во „Logic“ менито на Ледер алатникот.

### “RS-SR“ Флип-Флоп

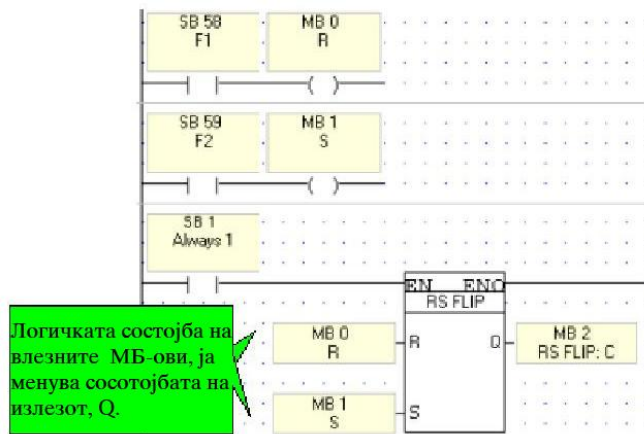
“RS-SR“ Флип-Флоп функциите се сместени во „Logic“ менито. Овие функции ја споредуваат логичката состојба на два влеза, и го употребуваат резултатот за да го одредат излезниот резултат во согласност со табелите прикажани подолу.

RS Флип-Флоп

R (A)	S (B)	Q
0	0	No change
0	1	1
1	0	0

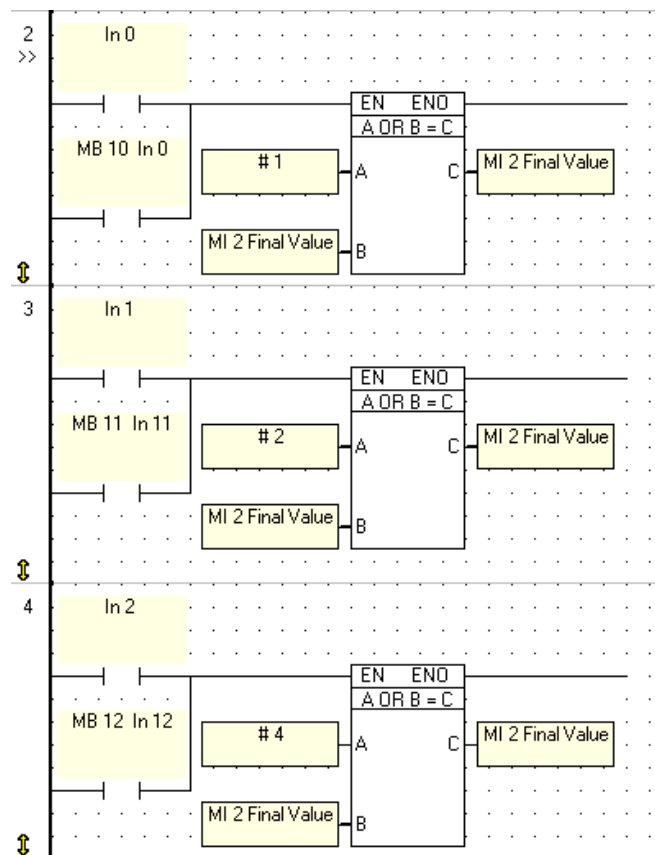
SR Флип-Флоп

S (A)	R (B)	Q
0	0	No change
0	1	0
1	1	1



## Бинарни Броеви

Мемориските цели броеви и системски броеви се 16-битни бинарни броеви. Кога ќе внесете декадни броеви за мемориските цели броеви и системските цели броеви, програмот ги конвертира во бинарни броеви и ја извршува назначената функција. Можна примена е при употреба на логичка функција за да се прикријат битови или да се провери дали има оштетен бит. Тоа може да се направи со употреба на декадни броеви кои се конвертираат во соодветни бинарни броеви. На следните слики е прикажана примена на декадните броеви {0,1,2,4,8,16,32,64,128, ...} кои се состојат од само една бинарна единица во логичка ИЛИ (OR) функција со цел да изврши прикривање на битови во влезните броеви во зависност од избирачката променлива.



$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	D
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	0	0	0	0	1	1	3
0	0	0	0	0	0	0	0	0	0	1	0	0	4
0	0	0	0	0	0	0	0	0	0	1	0	1	5
0	0	0	0	0	0	0	0	0	0	1	1	0	6
0	0	0	0	0	0	0	0	0	0	1	1	1	7
0	0	0	0	0	0	0	0	0	1	0	0	0	8
0	0	0	0	0	0	0	0	0	1	0	0	1	9
0	0	0	0	0	0	0	0	0	1	0	1	0	10
0	0	0	0	0	0	0	0	0	1	0	1	1	11
0	0	0	0	0	0	0	0	0	1	1	0	0	12
0	0	0	0	0	0	0	0	0	1	1	0	1	13
0	0	0	0	0	0	0	0	0	1	1	1	0	14
0	0	0	0	0	0	0	0	0	1	1	1	1	15
0	0	0	0	0	0	0	0	1	0	0	0	0	16
0	0	0	0	0	0	0	0	1	0	0	0	1	17
0	0	0	0	0	0	0	0	1	0	0	1	0	18
0	0	0	0	0	0	0	0	1	0	0	1	1	19
0	0	0	0	0	0	0	0	1	0	1	0	0	20
0	0	0	0	0	0	0	0	1	0	1	0	1	21
0	0	0	0	0	0	0	0	1	0	1	1	0	22
0	0	0	0	0	0	0	0	1	0	1	1	1	23
0	0	0	0	0	0	0	0	1	1	0	0	0	24
0	0	0	0	0	0	0	0	1	1	0	0	1	25
0	0	0	0	0	0	0	0	1	1	0	1	0	26
0	0	0	0	0	0	0	0	1	1	0	1	1	27
0	0	0	0	0	0	0	0	1	1	1	0	0	28
0	0	0	0	0	0	0	0	1	1	1	0	1	29
0	0	0	0	0	0	0	0	1	1	1	1	0	30
0	0	0	0	0	0	0	0	1	1	1	1	1	31

$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	D
0	0	0	0	0	0	0	1	0	0	0	0	0	32
0	0	0	0	0	0	0	1	0	0	0	0	1	33
0	0	0	0	0	0	0	1	0	0	0	1	0	34
0	0	0	0	0	0	0	1	0	0	0	1	1	35
0	0	0	0	0	0	0	1	0	0	1	0	0	36
0	0	0	0	0	0	0	1	0	0	1	0	1	37
0	0	0	0	0	0	0	1	0	0	1	1	0	38
0	0	0	0	0	0	0	1	0	0	1	1	1	39
0	0	0	0	0	0	0	1	0	1	0	0	0	40
0	0	0	0	0	0	0	1	0	1	0	0	1	41
0	0	0	0	0	0	0	1	0	1	0	1	0	42
0	0	0	0	0	0	0	1	0	1	0	1	1	43
0	0	0	0	0	0	0	1	0	1	1	0	0	44
0	0	0	0	0	0	0	1	0	1	1	0	1	45
0	0	0	0	0	0	0	1	0	1	1	1	0	46
0	0	0	0	0	0	0	1	0	1	1	1	1	47
0	0	0	0	0	0	0	1	1	0	0	0	0	48
0	0	0	0	0	0	0	1	1	0	0	0	1	49
0	0	0	0	0	0	0	1	1	0	0	1	0	50
0	0	0	0	0	0	0	1	1	0	0	1	1	51
0	0	0	0	0	0	0	1	1	0	1	0	0	52
0	0	0	0	0	0	0	1	1	0	1	0	1	53
0	0	0	0	0	0	0	1	1	0	1	1	0	54
0	0	0	0	0	0	0	1	1	0	1	1	1	55
0	0	0	0	0	0	0	1	1	1	0	0	0	56
0	0	0	0	0	0	0	1	1	1	0	0	1	57
0	0	0	0	0	0	0	1	1	1	0	1	0	58
0	0	0	0	0	0	0	1	1	1	0	1	1	59
0	0	0	0	0	0	0	1	1	1	1	0	0	60
0	0	0	0	0	0	0	1	1	1	1	0	1	61
0	0	0	0	0	0	0	1	1	1	1	1	0	62
0	0	0	0	0	0	0	1	1	1	1	1	1	63
0	0	0	0	0	0	1	0	0	0	0	0	0	64

## Математички Функции

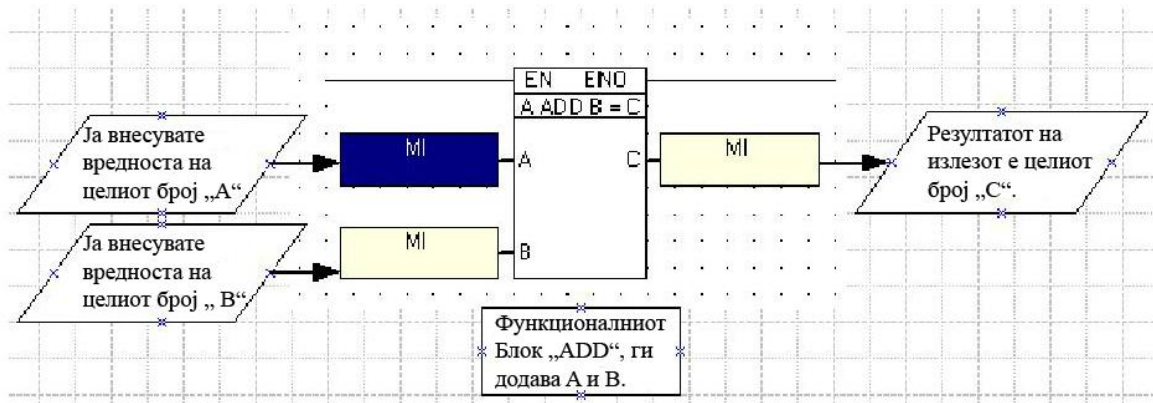
Во програмибилниот контролер може да се извршуваат математички пресметки со едноставно поставување на вградените математички функции на мрежа. Математичките функции кои се неѓаат во „Math“ менито овозможуваат:

- Зголемување/Намалување
- Собирање
- Одземање
- Множење
- Делење
- Коренување

- Степенување
- Факторизација
- Линеаризација

Секој тип на математички функции може да користи повеќе од 8 влезни вредности за да го пресмета резултатот. Внатрешната операција на функцискиот блок е достапна до корисникот.

Следниот пример прикажува функциски блок за собирање на два влезни операнди.



Операндите во листата која следи можат да се употребат за да обезбедат и влезни и излезни големини, со исклучок на константната вредност која може да обезбеди влезни вредности, но не може да содржи излезни вредности.

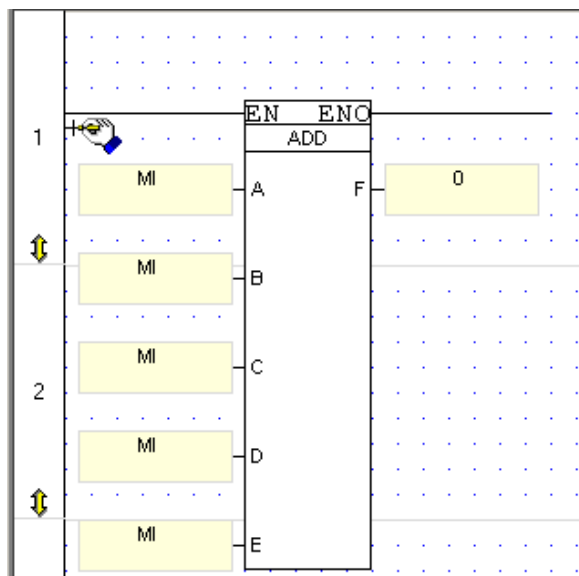
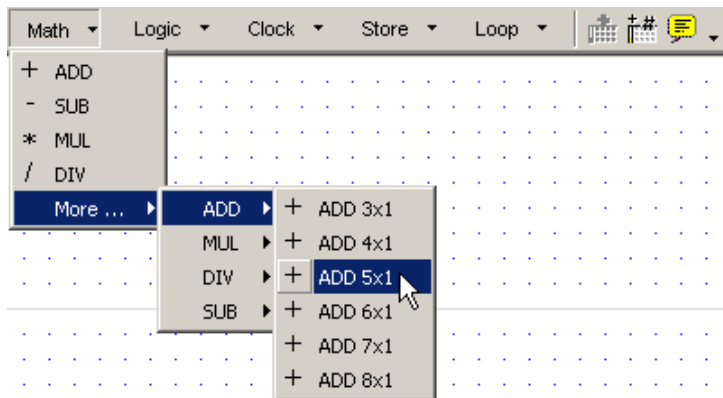
- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

### Повеќекратни влезни вредности во математичките функции

Може да се внесат до 8 вредности во математичките функциски блокови за функцијата да даде излезна вредност. На следниот пример е покажана функцијата „Add“ (собирање) која користи 5 влезни вредности. Секвенцата на поставување на функцискиот блок е следна:

1. Се кликува на копчето „Math“ од Ледер алатникот.
2. Десен-клик на Ледер мрежата за да се прикаже „Ladder pop-up“ менито.
3. Се избира „More“... и потоа се избира ја посакуваната функција.
4. Се кликува на функцијата со посакуваниот број од влезни големини.
5. Се поставува функцијата на посакуваната локација на мрежата. Мрежата автоматски се развлекува, за да ја смести функцијата.





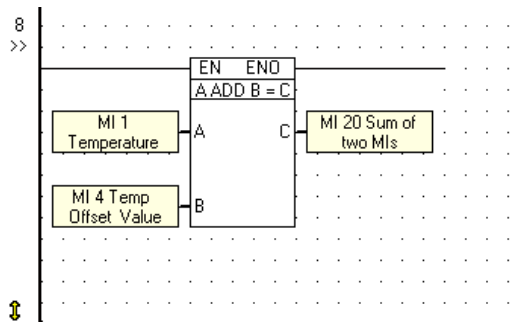
6. Се поврзуваат операндите со употреба на „Select Operand and Address“. Dialog box-от кој се отвара автоматски се додека сите влезни и излезни вредности не се поврзат.

### Собирање

Математичката функција собирање се извршува со функцискиот блок „Add“ прикажан подолу. Имате можност да додадете до 8 влезни вредности од следните операнди:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност. Примерот што следи ја прикажува функцијата „Add“, со две влезни вредности.

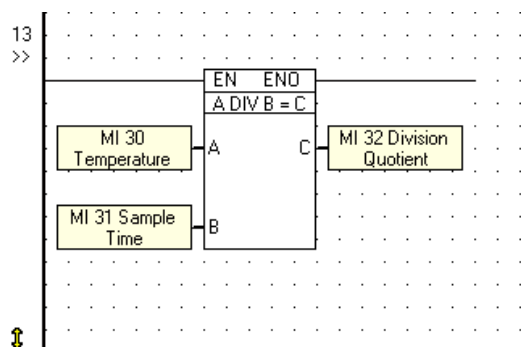


## Делење

Математичката функција делење се извршува со функцискиот блок „Divide“ прикажан подолу. Влезните вредности на оваа функција можат да бидат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди:(SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност.



Оваа функција може да се користи само за цели броеви. За делење на децимални броеви, се употребува функцијата „Divide“ од „Float“ менито.

Означените преостанати вредности се зачувани во SL 4, остаток при делење (Divide Remainder Signed) додека неозначените резултати се зачувани во SDW 4, остаток при делење (Divide Remainder Unsigned).

Се забележува дека мора да се зачуваат преостанатите вредности веднаш после функцијата за делење затоа што овие регистри ќе бидат повторно запишани од страна на наредната функција за делење.

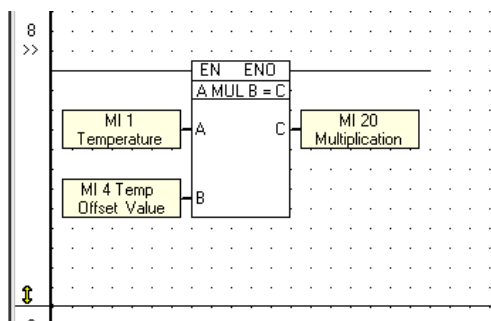
Вредностите не смеат да се делат со нула. Во случај да се случи ова, ќе се вклучи системскиот бит SB 4 – поделен со нула.

## Множење \*

Математичката функција множење се извршува со функцискиот блок „Multiply“ прикажан подолу. Може да се помножат до 8 влезни вредности од типот:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди (SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност. На следната слика е прикажана функцијата множење, со две влезни вредности.

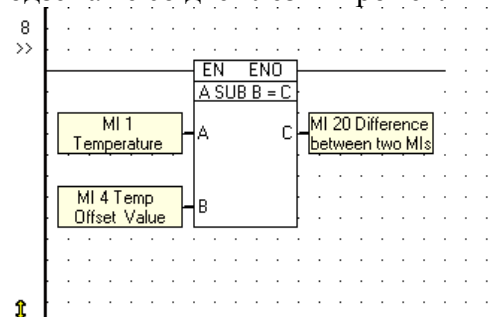


## Одземање -

Математичката функција одземање се извршува со помош на функцискиот блок „Subtract“ прикажан подолу. Функцијата се наоѓа во менито „Math“. Влезни вредности на функцијата одземање можат да бидат:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди (SI) (SL)(SDW)
- Мрежен Системски Цел Број (NSI)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност. На следната слика е прикажана примена на функцискиот блок одземање со две влезни променливи.



## Линеаризација, Векторска Линеаризација

Функциите за линеаризација, кои се наоѓаат во „Math“ менито, овозможуваат конверзија на вредности. Можете да се искористат на пример за конверзија на аналогни влезни вредности во реални мерни единици како степени целзиусови.

### Линеаризација на единечна вредност

Оваа функција ја линеаризира единечната влезна вредност која потоа ја зачувува во посакуваниот регистар. Ако, на пример, X1 и Y1 се 0, и X2=100 а Y2=1023, излезната вредност ќе се линеаризира како на графиконот.

The diagram illustrates the linear conversion process. On the right, a graph shows a straight line starting from the origin (0,0) and extending to the point (100, 1023). Two intermediate points are marked: (x1, y1) and (x2, y2), with dashed lines indicating their coordinates on the axes.

On the left, a software interface shows the configuration of the Linear function. The main window displays a table of parameters:

Func	Operand	Address	Format	Description
X1	M1	50	DEC	Linear conversion: X1 Value
Y1	M1	51	DEC	Linear conversion: Y1 Value
X2	M1	52	DEC	Linear conversion: X2 Value
Y2	M1	53	DEC	Linear conversion: Y2 Value
X	M1	54	DEC	Linear conversion: X (input) Value
Y	M1	55	DEC	Linear conversion: Y (result) Value

Below this table, a smaller dialog box shows the configuration for the output register Y:

Direct: [M1] 55 Linear conversion: Y (result) Value [DEC]

Green callouts explain the parameters: "Овие се параметрите кои функцијата ги користи, за да ги конвертира влезните вредности." (These are the parameters the function uses to convert the input values.), "Ова е вредноста, која треба да се конвертира." (This is the value to be converted.), and "Ова е резултатот." (This is the result.).

Овие вредности ќе предизвикаат:

Температурниот влез од 1000° C да се конвертира во 1023 Дигитална вредност.

Температурниот влез од 500° C да се конвертира во 512 Дигитална вредност.

### Линеаризација на Векторот на Вредностите

Оваа функција го линеаризира векторот на влезната вредност и потоа ја зачувува вредноста во посакуваниот вектор.

The screenshot shows the "Vector of Values: Linearization" dialog box. It contains a table with the following data:

Params	Func	Operand	Address	Format	Description
IN	X1	M1	0	DEC	Linear conversion: X1 Value
	Y1	M1	1	DEC	Linear conversion: Y1 Value
	X2	M1	2	DEC	Linear conversion: X2 Value
	Y2	M1	3	DEC	Linear conversion: Y2 Value
	Xs	M1	4	DEC	Start address : Source Vector
	Length	M1	5	DEC	Vector length (max.=128 registers)
OUT	Ys	M1	6	DEC	Start address : Target Vector

Buttons at the bottom include OK, Cancel, and Help.

Можете да конвертирате вредности содржани во следниве типови на операнди:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди (SI) (SL)(SDW)

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност од блокот за линеаризација.

### Линеаризација на Аналогни I/O вредности.

Аналогните излезни вредности мораат да бидат содржани во регистарот кој го поврзувате со излезот во хардверската конфигурација.

MI 13 е поврзан со аналоген 4-20mA тип на излез.

Функцијата од линеаризираниот Ф.Б. е поврзана со истиот операнд, кој ја содржи вредноста на аналогниот излез.

Params	Func	Operand	Address	Format	Description
		D#	0		DEC Linear conversion: Y1 Value
		D#	819		DEC Linear conversion: X2 Value
		D#	5000		DEC Linear conversion: Y2 Value
		D#	4095		DEC Linear conversion: X1 Value
		MI	12		DEC mBar 0-5000
OUT	Y	MI	13		DEC 12-bit Analog Output IO-AI4-AO4

### Работење во граници од 4-20mA

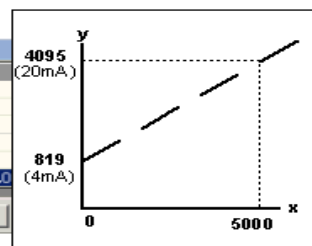
Се забележува дека уредите кои се користат во конјукција со контролерот мора да се калибрираат секој посебно. Во примерот кој следи, аналогниот уред е притисен сензор, затоа вредностите се претвараат од милиампери во милибари.

### 10-битен Аналоген влез, V200-18-E1

Params	Func	Operand	Address	Format	Description
IN	X1	D#	204		DEC Linear conversion: X1 Value
	Y1	D#	0		DEC Linear conversion: Y1 Value
	X2	D#	1023		DEC Linear conversion: X2 Value
	Y2	D#	5000		DEC Linear conversion: Y2 Value
	X	MI	11		DEC 10-bit Analog Input: V200-18-E1
OUT	Y	MI	12		DEC mBar 0-5000

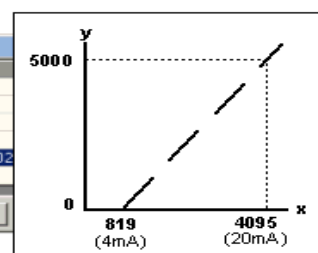
## 12-битен Аналоген Излез, IO-A14-AO2

Linearization							
Params	Func	Operand	Address		Format	Description	
IN	X1	D#	0		DEC	Linear conversion: X1 Value	
	Y1	D#	819		DEC	Linear conversion: Y1 Value	
	X2	D#	5000		DEC	Linear conversion: X2 Value	
	Y2	D#	4095		DEC	Linear conversion: Y2 Value	
OUT	X	MI	12		DEC	mBar 0-5000	
	Y	MI	13		DEC	12-bit Analog Output IO-A14-AO2	



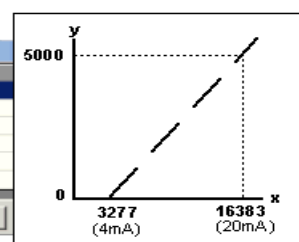
## 12-битен Аналоген влез, IO-A14-AO2

Linearization							
Params	Func	Operand	Address		Format	Description	
IN	X1	D#	819		DEC	Linear conversion: X1 Value	
	Y1	D#	0		DEC	Linear conversion: Y1 Value	
	X2	D#	4095		DEC	Linear conversion: X2 Value	
	Y2	D#	5000		DEC	Linear conversion: Y2 Value	
OUT	X	MI	14		DEC	12-bit Analog Input: IO-A14-AO2	
	Y	MI	15		DEC	mBar	



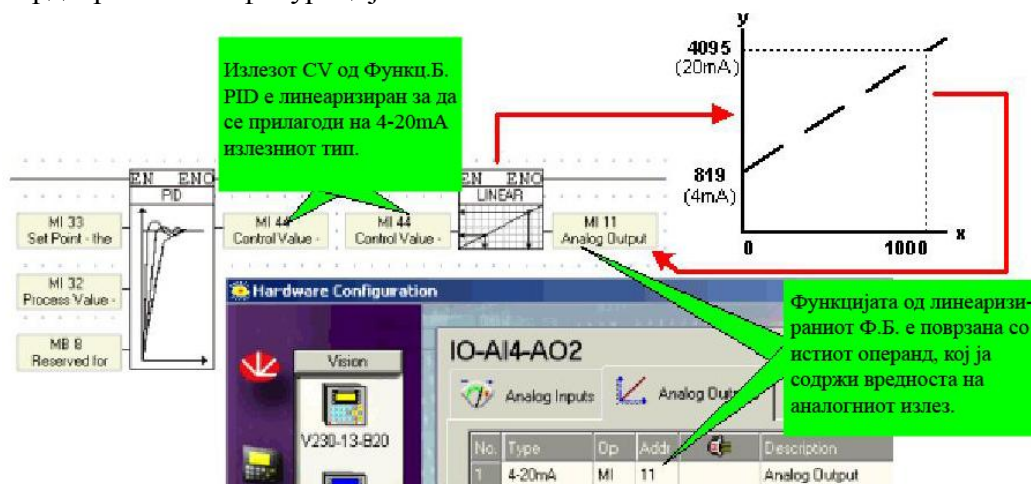
## 14-bit Аналоген влез, V120-12-UN2

Linearization							
Params	Func	Operand	Address		Format	Description	
IN	X1	D#	3277		DEC	Linear conversion: X1 Value	
	Y1	D#	0		DEC	Linear conversion: Y1 Value	
	X2	D#	16383		DEC	Linear conversion: X2 Value	
	Y2	D#	5000		DEC	Linear conversion: Y2 Value	
OUT	X	MI	16		DEC	Vision 120 AI 4-20mA/14 bit	
	Y	MI	17		DEC	mBar	



## Линеаризација на PID Аналогна Излезна Вредност

Аналогните вредности можат да бидат конвертирани во физички мерни единици, како степени Целзиусови со користење на линеаризациски функциски блок. Аналогните излезни вредности се содржани во регистарот кој се поврзува со излезот во хардверската конфигурација.



## Линеаризација на PID излезот во аналоген излез

Достапниот опсег во согласност со контролерот и влезно излезниот модул е прикажан во поглавјето Аналогно I/O подрачје. Треба да се забележи дека уредите кои се користат во конјукција со контролерот мораат да се калибрираат секој посебно. Границите можат да бидат поставени на излезот, во овој случај линеаризација не е потребна.

**PID Parameters:**

IN	Type	Value	Description
SpPv-high	MI	89	Process Value high limit - the maximum PV input value
SpPv-low	MI	80	Process Value low limit - the minimum PV input value
Cv-High	D#	4095	Control Value high limit - the maximum CV output value
Cv-Low	D#	819	Control Value low limit - the minimum CV output value
Reserved	MI	43	Reserved for future use
Reverse	MB	6	Reverse action: 1: Reverse action, 0: Direct action
RST Intgl	MB	7	Reset integral accumulated error; 1: Clear, 0: Continue
Reserved	MB	8	Reserved for future use
CV	MI	11	Control Value - the PID output
CV(k)	MI	45	Control Value k <sub>p</sub> result
CV(i)	MI	46	Control Value i <sub>l</sub> result
CV(d)	MI	47	Control Value d <sub>l</sub> result

**Hardware Configuration:**

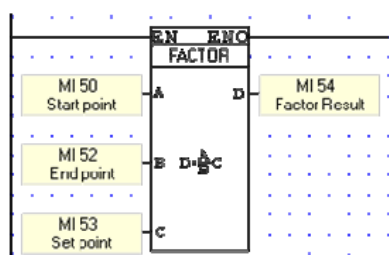
No.	Type	Addr.	Description
1	4-20mA	MI 11	Control Value-the PID output
2	None		

## Факторизација

Математичката функција Фактор користи 3 влезни вредности. Фактор ја дели влезната вредност А со влезната вредност В и потоа го множи резултатот со влезната вредност С. Резултатот е зачуван во излезниот операнд D. Можат да се користат следниве типови на операнди во оваа операција:

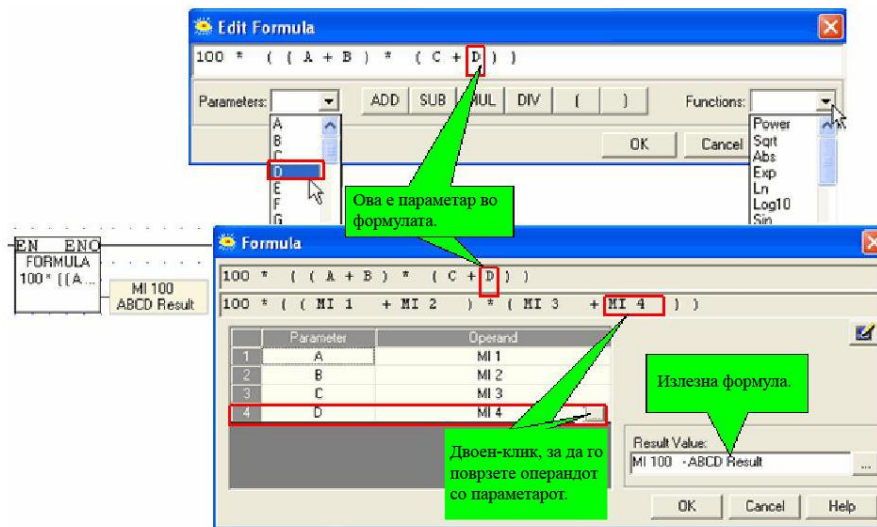
- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди: (SI) (SL) (SDW)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност. Ви примерот кој што следи е покажана функцијата Фактор.

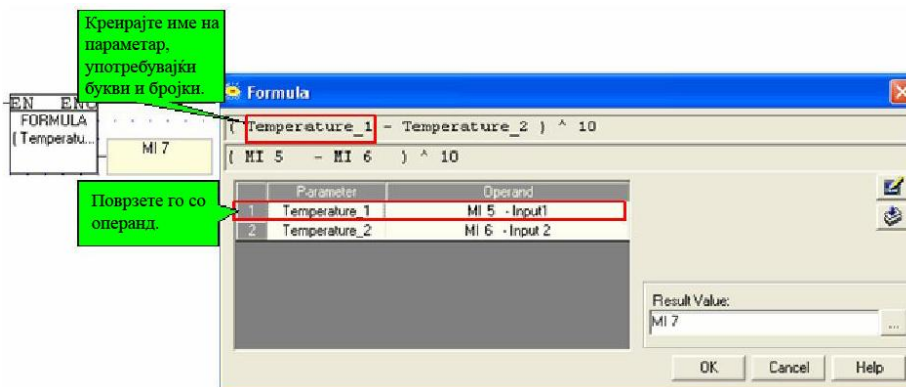


## Формула: Создадете Сопствена

Функцијата Формула, која се наоѓа на менито „Math“, ви овозможува да употребите математички оператори во вредностите на операндот, и да добиете излезен резултат во регистарот. За да се креира формула се поставува функцијата формула на лидер мрежата при што ќе се отвори „Edit formula“. Можат да се внесат константни броеви, параметри и оператори. Исто така можат да се изберат параметри и оператори од листата. Синтаксата на формулата одговара на нормална математичка ознака. Со исклучок на знакот минус, бинарните оператори не можат да бидат употребени за да се започне формула. Останатите бинарни оператори вклучуваат Собирање [+], Множење [\*], Делење [/], Загради [( )], Степен (Power). Единечни оператори како синус можат да се употребат за да се започне формула.



Име на параметар се креира со користење на комбинација од букви и знаци.



Името на параметарот не може да започне со бројка или да содржи празно место. Наместо празно место се употребува знакот долна линија (\_). Излезната константа не смее да ја надмине вредноста на MF или ML. Во следните случаи контролерот ќе ја процесира формулата употребувајќи движечки регистри:

- Ако формулата содржи еден или повеќе движечки операнди.
- Ако константната вредност во формулата не е цел број
- Ако операторот, како тригонометриските оператори, има потреба PLC-то да користи движечки регистри за да ја заврши операцијата.



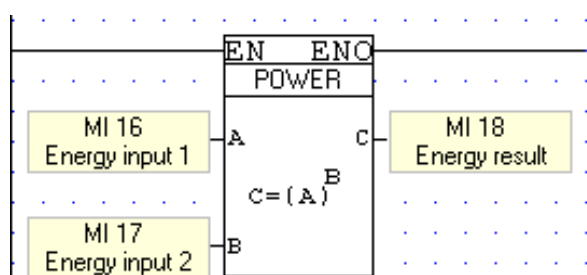
## Степен (Power)

Математичката функција степен користи две влезни големини. Функцијата степен, ја зголемува влезната вредност А со степенот на влезната вредност В (експонент). Резултатот е зачуван во излезниот операнд С. Функцијата се наоѓа во менито „Math“.

Во оваа операција можат да се употребат следниве типови на операнди:

- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди (SI) (SL)(SDW)
- Константна Вредност #

Со исклучок на константната вредност, било кој од овие операнди можат да содржат излезна вредност. Следниот пример ја прикажува функцијата степен.

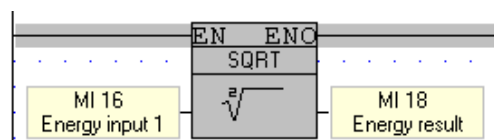


## Квадратен Корен (Square Root)

Како резултат од оваа функција се добива квадратниот корен на влезната вредност. Резултатот се зачувува во излезниот операнд. Функцијата се наоѓа во менито „Math“. Може да се пресметува квадратен корен на вредностите содржани во следниве типови на операнди:

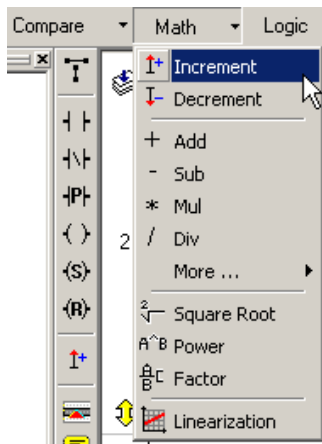
- Мемориски Цел Број (MI)
- Мемориски Цел Број (со повеќе места) (ML)
- Двоен Слог (DW)
- Системски Операнди (SI) (SL)(SDW)

Следниот пример ја прикажува функцијата квадратен корен.

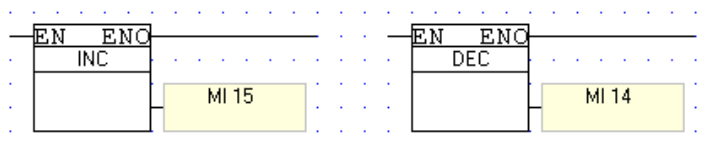


## Зголемување/Намалување

Двете функции се наоѓат на функциското мени „Math“; копчето „Increment“ исто така се наоѓа на „shortcut“ алатникот.

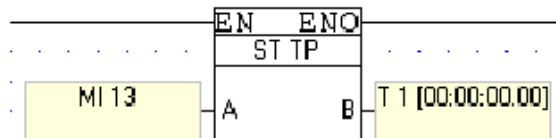


„Increment“ ја зголемува вредноста на селектираниот операнд за 1.  
 „Decrement“ ја намалува вредноста на селектираниот операнд за 1.



### Store Timer/Counter

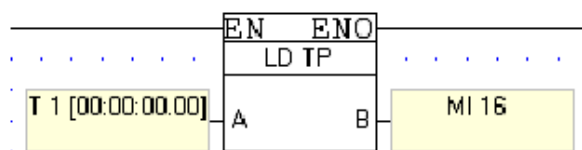
Со оваа функција се извршува меморирање на еден операнд во вредност на тајмер или бројач.



Операндот MI 13 ја содржи вредноста што треба да биде меморирана во друга вредност, излезот B го содржи операндот T1 добиената вредност.  
 На пример ако MI 13 е еднаков на 1023 меморираната вредност ќе биде 10 секунди и 23 милисекунди.

### Load Timer/Counter

Со оваа функција се извршува отчитување на вредноста на тајмер или бројач во друг операнд.

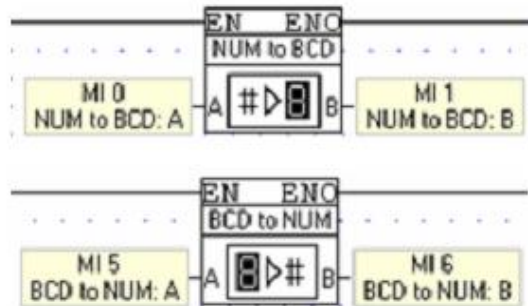


На влезот A се доведува вредноста на тајмерот или бројачот, на излезот B се добива вредност која се запишува во соодветен операнд.

На пример ако T1 е еднакво на 10 секунди и 23 милисекунди добиената вредност MI 16 ќе биде 1023.

### BCD во NUM и NUM во BCD

Со оваа функција може да се конвертираат нумерички вредности во BCD (Binary CodeD) или BCD во нумерички вредности.



### Clock функција

Clock функцијата ги содржи следниве наредби:



- Време



- Ден од неделата



- Ден од месецот



- Месец

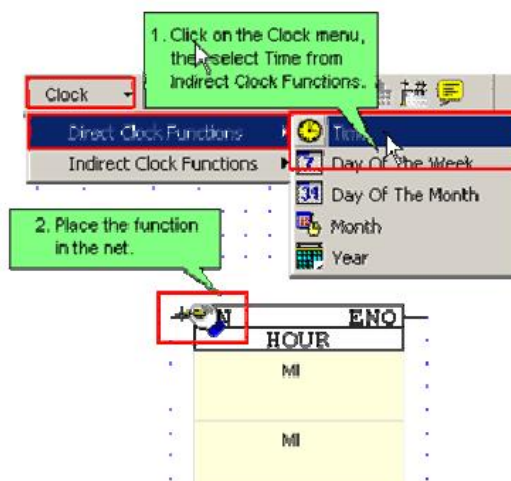


- Година

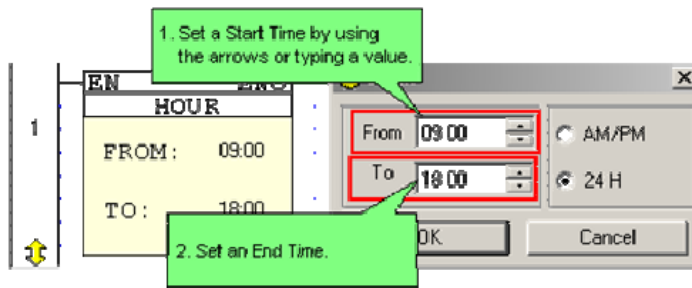
Пример основан на овие наредби:

Сакаме да се вклучува некој мотор од 9.00 часот до 18.00 часот, од понеделник до петок, почнуваќи од 15 во месецот па се до 24, во 2001 и 2002 година

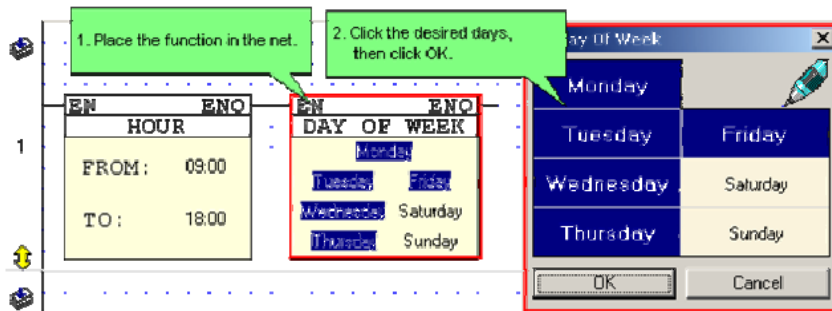
1) Се поставува функцијата во мрежата



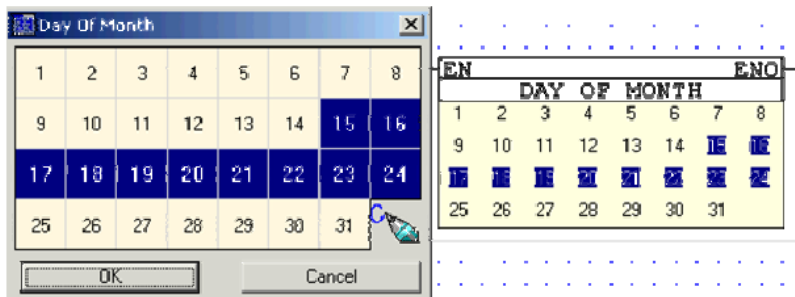
2) Се одредува почетокот и крајот на времето



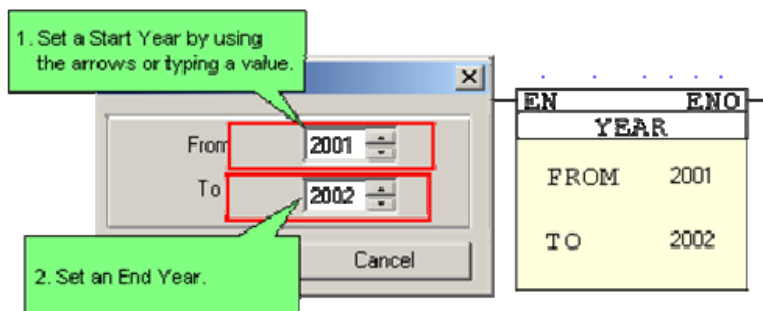
3) Се селектираат деновите во неделата



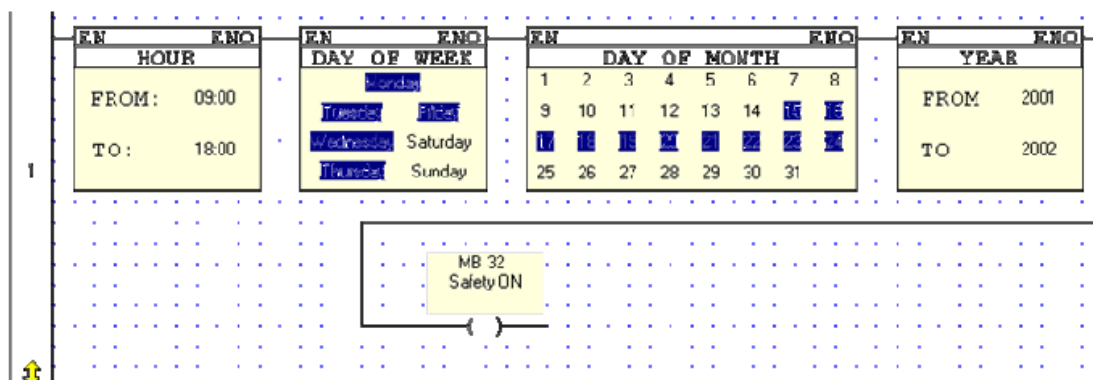
4) Се селектираат деновите во месецот



5) Се селектира годината



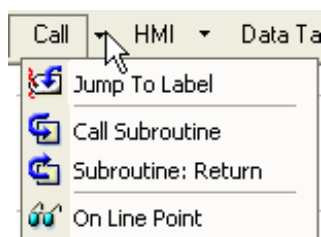
## 6) Изглед на поврзаноста на елементите во мрежата



## Повикувања: Контрола на Програмот

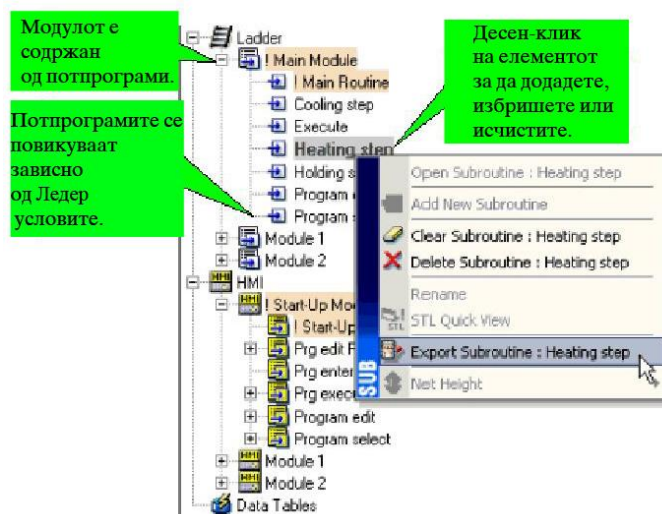
### Повикувања, Скокови и Ознаки

Функциите од „Call“ менито ви овозможуваат да го подготвите редоследот според кој вашата програма се стартува.




## Секвенци на Програмот: Модули, Потпрограми, Ознаки и Скокови.

Модулот се содржи од потпрограми. Модулите и потпрограмите се употребуваат за да ја разделите вашата апликација во програмски блокови. Тогаш ќе можете да ги стартувате овие програмски блокови од било која позиција во контролната апликација.

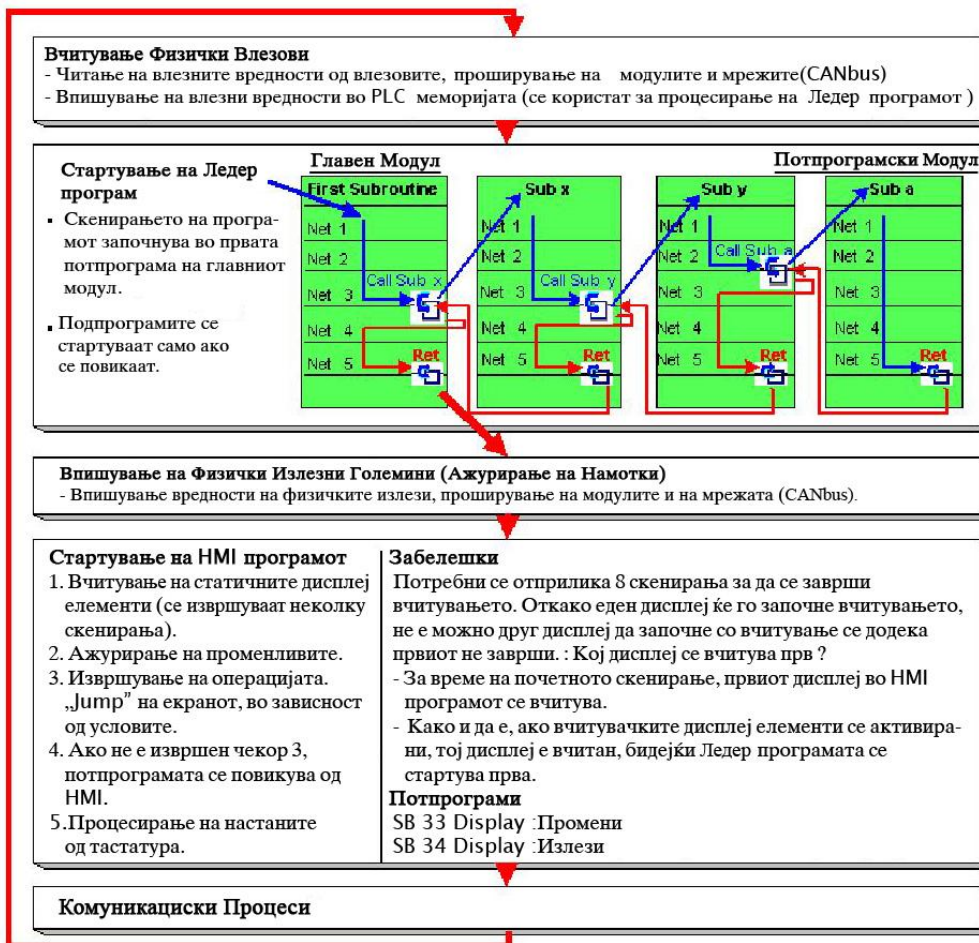


Во програмското мени елементите се распределени по азбучен ред. Тоа не влијае на редоследот на стартување на програмата. Ледер модулите и потпрограмите можат да се задвижат според начинот „повлечи и испушти“, како HMI модулите и дисплеите. Главниот Ледер модул, главната потпрограма, Start-up HMI модул и Start-up HMI дисплејот не можат да се задвижат според начинот „повлечи и испушти“ или да бидат избришани. За лесна идентификација, тие секогаш се во портокалова боја. За да се контролира програмскиот тек и да се избегне повторувањето, се употребува функцијата за повикување потпрограма од повикувачките потпрограми. Во потпрограмата, го контролирате текот постојано прескокнувајќи преку мрежите, употребувајќи ознаки и скокови во Ледер функциите. Тоа овозможува да се добие во време.

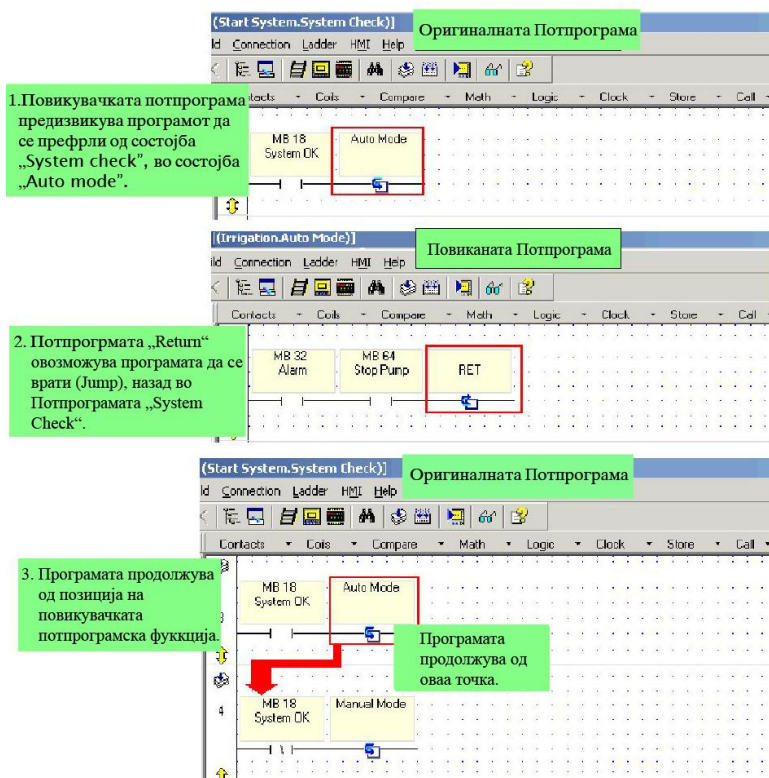
Новиот VisiLogic проект ги содржи, главниот модул и потпрограма на програмот. Секоја нова потпрограма содржи одреден број на мрежи и потпрограмски повратни функции. Потпрограмите не се стартуваат ако не се повикаат од повикувачката потпрограма. Ако ниту една повикувачка потпрограма не е вклучена во првата потпрограма на главниот модул, програмата се стартува се додека не ја достигне потпрограмската повратна функција, и тогаш се враќа назад на почетокот на првата потпрограма. Ако потпрограмата не се стартува, намотките во потпрограмата нема да се ажурираат. На пример, Потпрограмата 4 содржи . Ако MB0 е вклучен во потпрограмата 1, но потпрограмата 4 не е повикана 00 не е ажурирано. Начинот според кој I/O се ажурираат, зависи од скенирањето на PLC програмот. Некои од функционалните блокови имаат потреба од конфигурирање. Конфигурирањето треба да се оствари во првиот потпрограм на главниот програмски модул. Ако конфигурирањето е во потпрограмата која не е повикана од програмот поврзаните програмски блокови нема да се процесираат иако условите за активирање на функционалните блокови се активни. Потпрограмите можат да се употребат онолку пати колку што има потреба. Потпрограмите исто така можат да се користат и помеѓу проектите.

## Скенирање на PLC Програмата

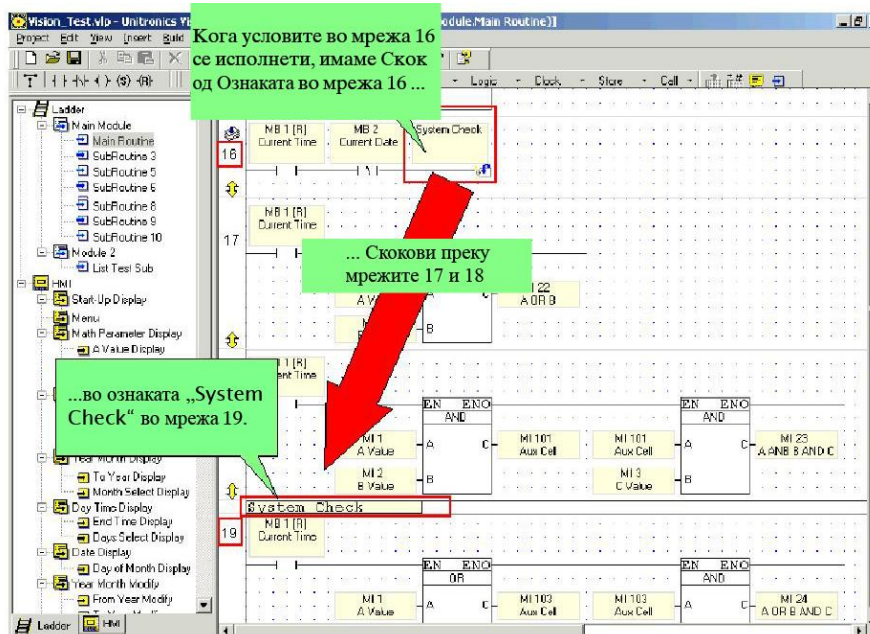
Скенирањето се извршува комплетно од контролерите на целата програма. Циклусот на скенирање се изведува континуирано. Задачите за вклучување се однесуваат на состојбата на SB2 вклучувачкиот bit, и се изведуваат кога контролерот е вклучен. Овие функции се извршуваат пред скенирањето на програмот. Времето за скенирање е зачувано во: „SI 0 Scan Time, Resolution: Units of 10 mSec“.



## Повикување Потпрограма и Потпрограма: Враќање



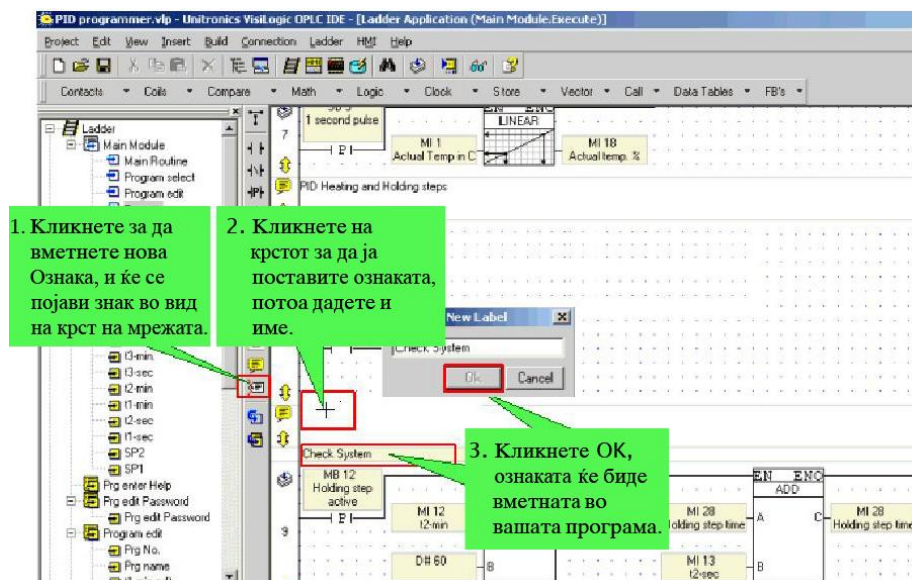
## Во Програмите: Ознаки и Скокови



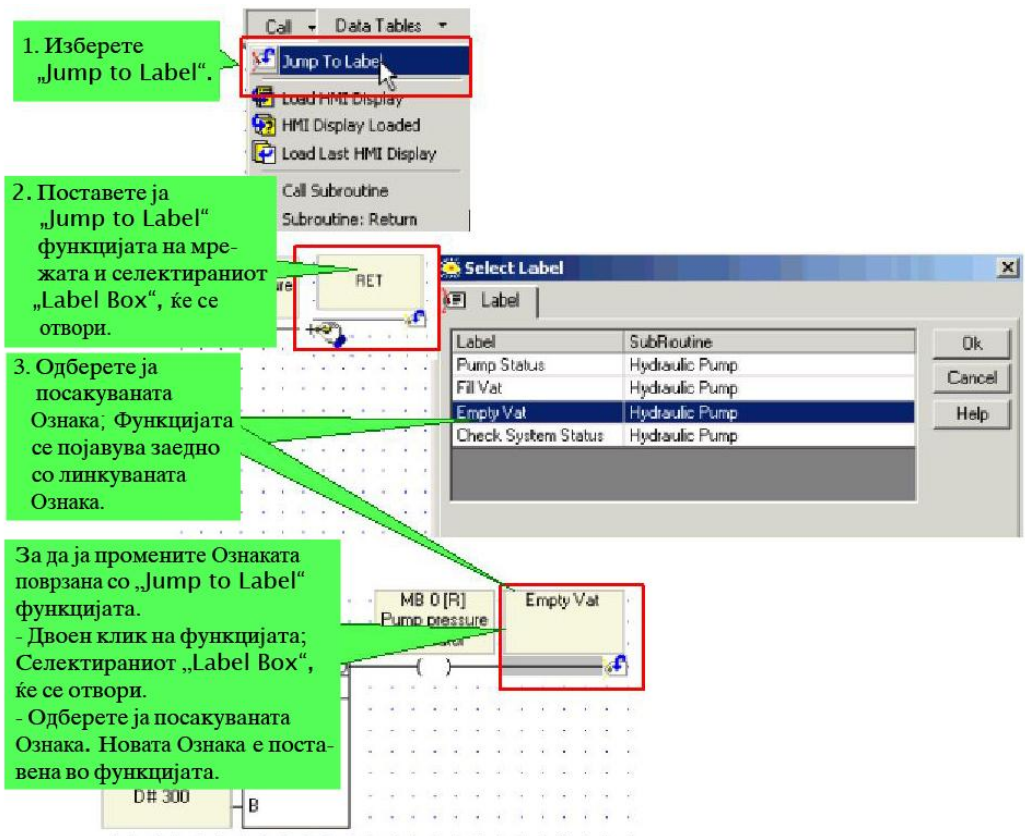
## Ознаки и Скокови

Ознаките (Labels) овозможуваат да скокнете (Jump) преку лидер мрежите, внатре во потпрограмата. Употреба на Ознаките:

1. Поставете ознака во мрежата.
2. Создадете услов за прескокнување.
3. Поставете го прескокнувањето, после условот.

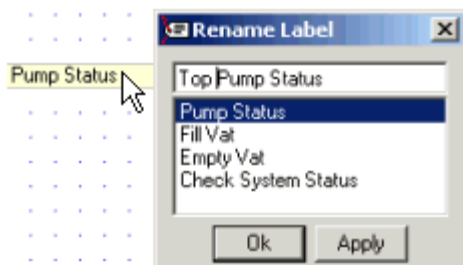






## Преименување на Ознаки

За да го промените името на ознаката, направете двоен клик на неа, внесете го новото име и кликнете „Apply“.

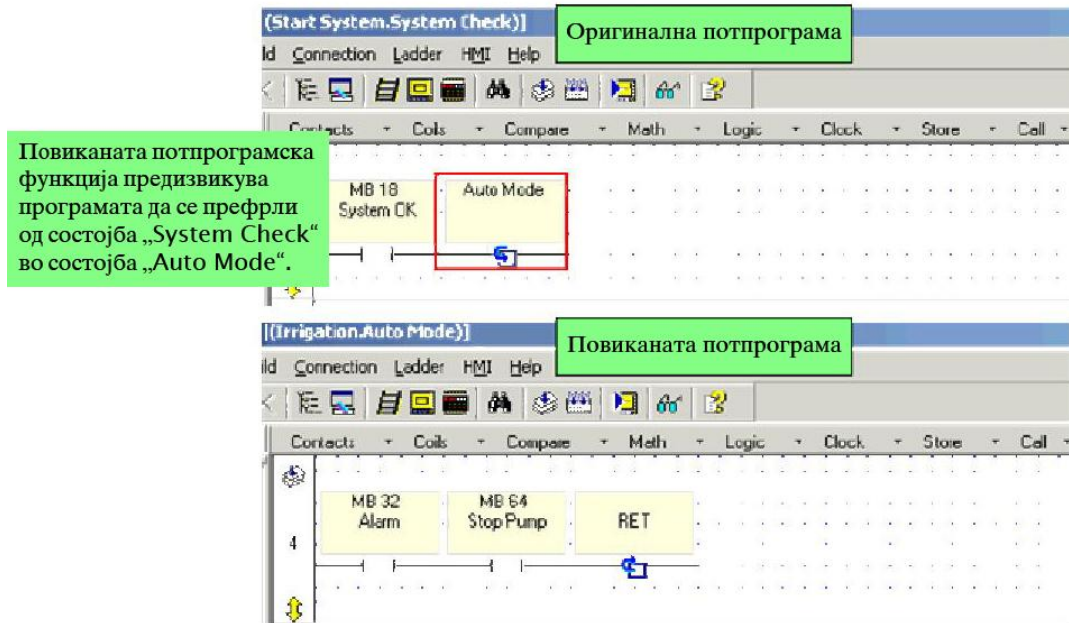


Исто така можете да уботребувате ознаки како „bookmarks“, употребувајќи ги за да обележите програмски делови и тогаш лоцирајте ги, употребувајќи:

Оди во Ознаката <Alt> + <Right/Left arrow> и листата на Ознаки <Ctrl> + <L>.

## Повикување Потпрограма

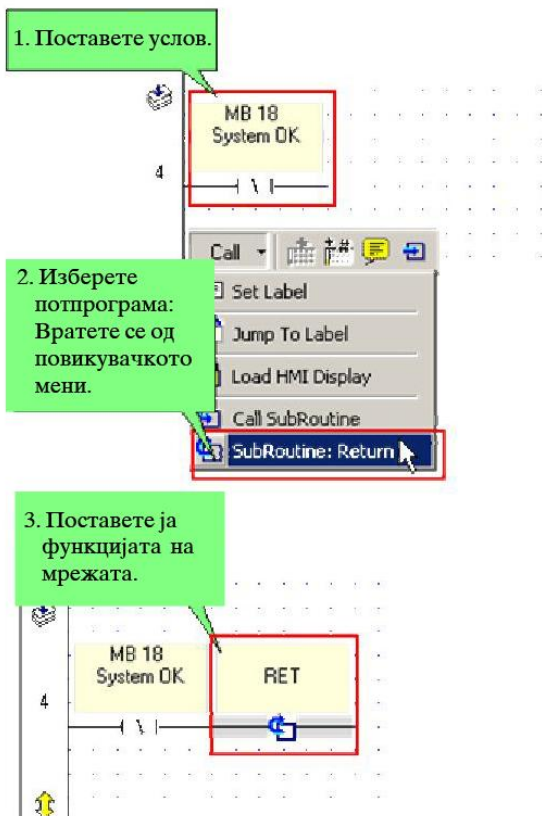
Оваа функција предизвикува потпрограмата да се стартува во зависност од Ледер условите.



## Употреба на повикувачка потпрограма

### Потпрограма: Враќање

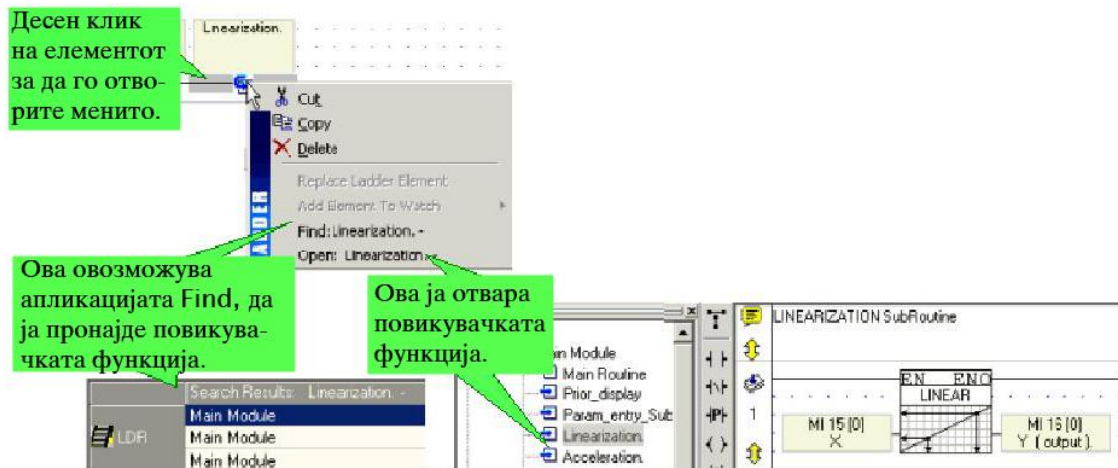
Потпрограмата се стартува се додека не ја достигне потпрограмската повратна функција, и потоа се враќа на почетокот на претходната потпрограма. Употребување на потпрограмско враќање



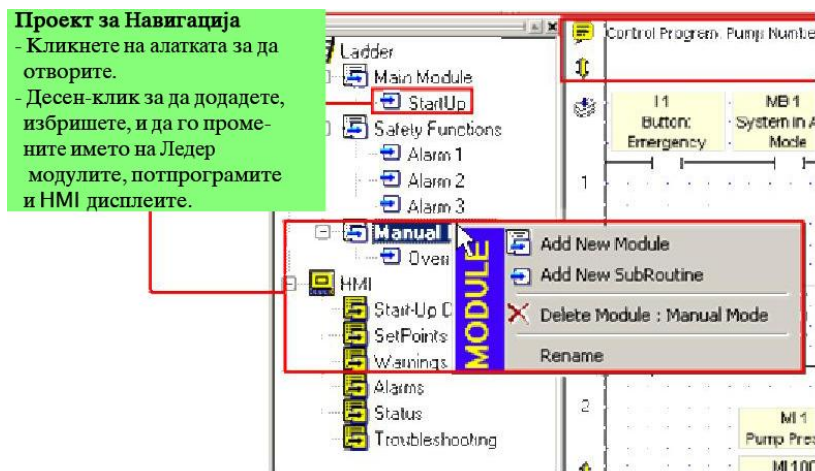
## Отворање Потпрограма

За да отворите Потпрограма и да додадете:

- Двоен клик на „Project Explorer“ менито,
- Десен клик на Потпрограмата во „Project Explorer“ менито, потоа изберете „Open“,
- Десен-клик на елементот на повикувачката Потпрограма за да пристапите на посакуваната Потпрограма.



## Именувај-Промени име на Модули и Потпрограми



## Моментални Елементи

Моменталните Елементи можете да ги најдете на „More> Immediate“ менито. Генерално, I/O вредности се вчитуваат и впишуваат во согласност со скенирањето на PLC програмот. Моменталните Елементи веднаш ја ажурираат тековната вредност на I/O--без да се обрне внимание на скенирањето на програмата.

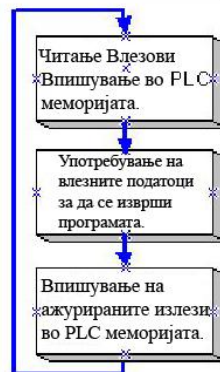
Тоа ве оневозможува да впишувате вредности на влезовите, да ја употребите новата влезна вредност за да ја довршите PLC програмата, и да ги вклучите излезите, како на пример кога тоа е итно потребно.

Ако вашиот програм бара веднаш да внесете I/O вредност, употребете ги моменталните елементи за да ги поврзете со прекинувачката рутина.

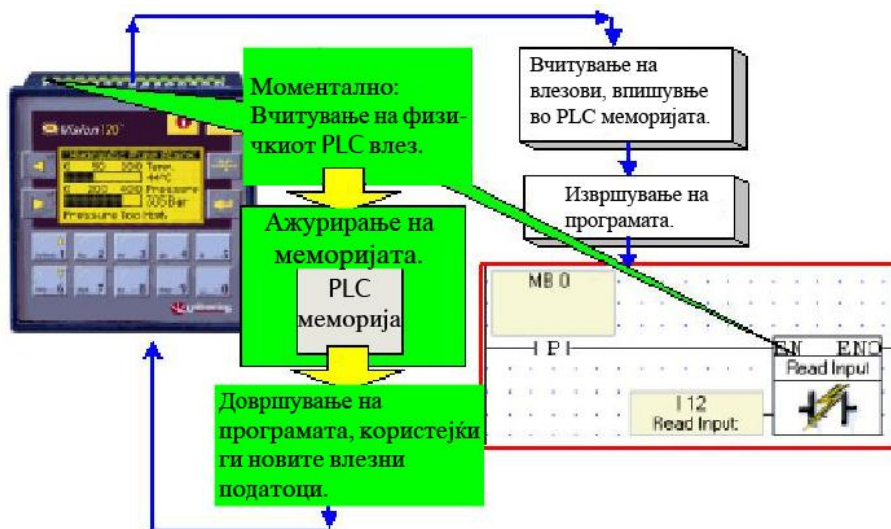
## Моментално: Читање на Физички Влез

Читање на физички влез ( Read Physical Input ) се наоѓа во „More> Immediate“ менито. Овој елемент може да се употреби за читање на моменталниот статус на физичкиот, хардверскиот влез и го користи новиот влезен статус за да ја изврши PLC програмата. Вообичаено, скенирањето на PLC се стартува на следниот начин:

Скенирање на Програмот



Кога програмата наидува на „читање на физичкиот влез“, програмата моментално го чита физичкиот PLC влез, ја ажурира PLC меморијата, и го извршува остатокот од програмата, користејќи ги новите влезни податоци.



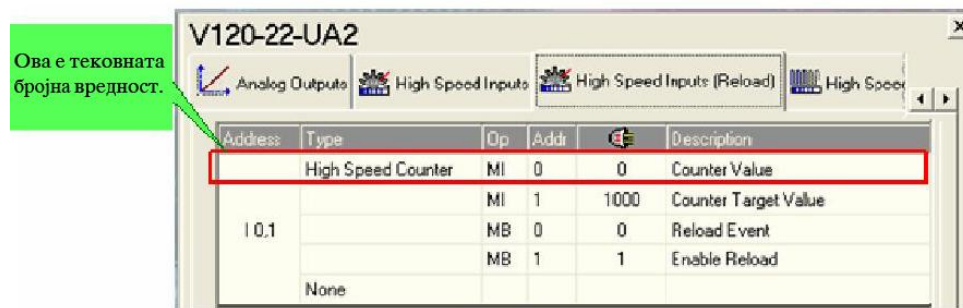
За да го употребите читањето на физичкиот влез, поставете го на мрежата после активирачките услови и изберете го посакуваниот влез. Во мрежата, „читање на физички влез“ треба да остане сам, освен во случај на активирање.

## Моментално: Ажурирање на Високо-Брзинскиот Влез

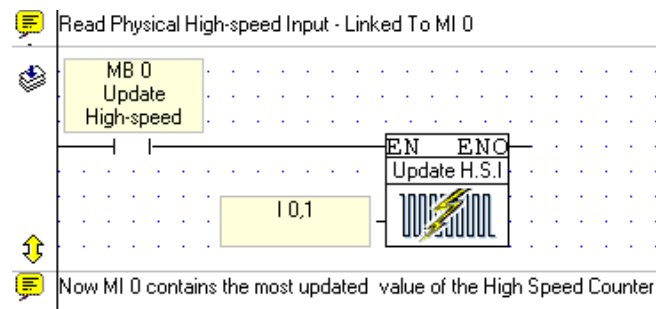
„Ажурирање на високобрзинскиот влез“ се наоѓа во „More> Immediate“ менито. Овој елемент може да се употреби за моментално ажурирање на тековната вредност на физичкиот, хардверскиот високо-брзински влез без да се обрне внимание на скенирањето на програмот и да се искористи новата влезна големина за да се доврши PLC програмот.



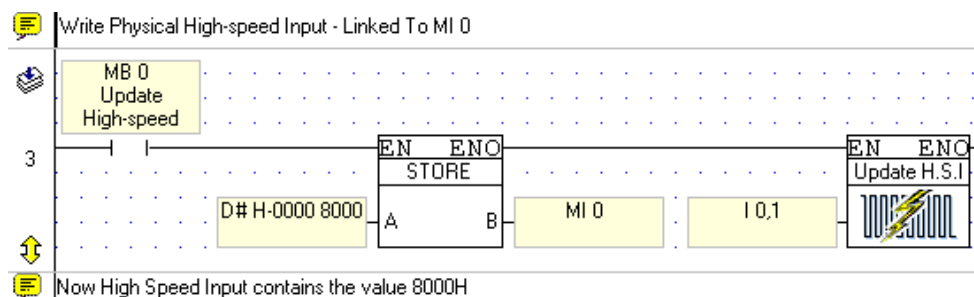
Кога програмот ќе најде на „ажурирање на високобрзински влез“, програмот веднаш ја споредува вистинската тековна Влезна вредност со вредноста во MI поврзана со влезот.



Ако вредностите не се еднакви, MI е ажуриран со тековната влезна вредност; и остатокот од програмот се довршува според новите влезни податоци. За да го употребите „ажурирање на високобрзински влез“, поставете го на мрежа откако сте ги активирале условите и изберете го посакуваниот влез.



Внатре во мрежата, „ажурирање на високобрзински влез“ треба да остане сам, освен во случај на активирање.



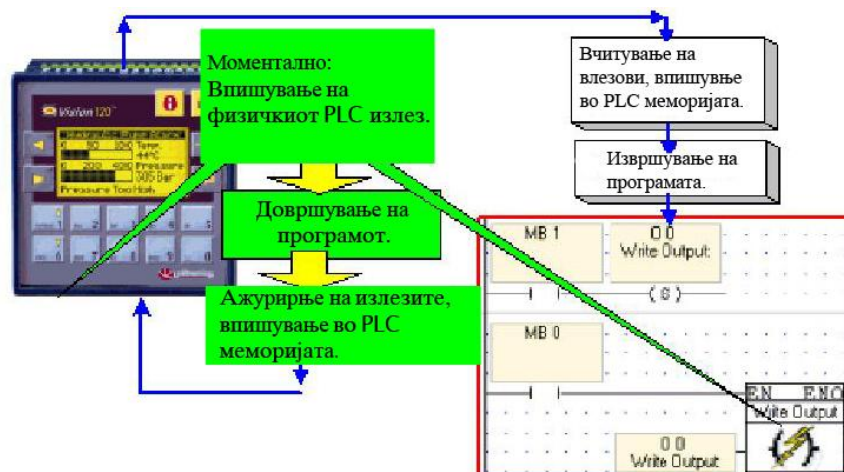
## Моментално: Впишување на Излез

Впишување на излез (Write to Output) се наоѓа во „More> Immediate“ менито. Овој елемент може да се употреби за директно ажурирање на статусот на физичкиот, хардверски излез. Вообичаено, скенирањето на PLC се стартува на овој начин.

Скенирање на Програмот



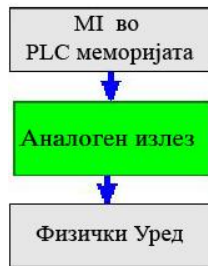
Кога програмот наидува на „Write to Output“, програмот моментално го запишува физичкиот PLC излез, потоа го довршува остатокот од програмата.



За да го употребите „Write to Output“, поставете го на мрежа откако ќе ги поставите условите и изберете го посакуваниот излез. Внатре во мрежата, „впишување на излез“ треба да остане сам освен во случај на негово активирање. Ако, откако „впишување на излез“ е извршен, истиот излез е ажуриран и додека остатокот од програмата се стартува, последното ажурирање е испишано во PLC меморијата на крајот од скенирањето на програмот.

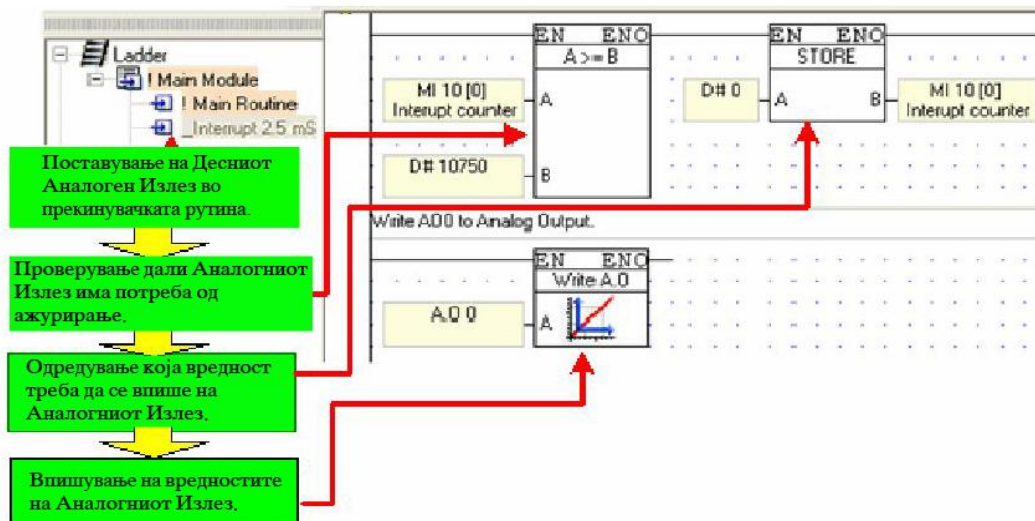
## Моментално: Впишување на Физичкиот Аналоген Излез

Впишување на физичкиот аналоген излез се наоѓа во „More> Immediate“ менито. Овој елемент може да се употреби за моментално вна вредноста на физичкиот, хардверски излез, без да се обрне внимание на скенирањето на програмот. Оваа функција генерално е вклучена во прекинувачката рутина, како на пример, излезот да се вклучи во случај на алармантна состојба. Во мрежата, „впишување на физичкиот аналоген излез“ треба да остане сам.



Кога моменталното впишување е повикано, вредноста на поврзаниот MI веднаш е впишана на физичкиот аналоген излез

No.	Type	MI	Addr	Q	Description
0	0-10V	MI 0	0	0	Value to A0
1	None				
2	None				
3	None				



## 5. Опис на програмибилен контролер од австрискиот производител V&R

Контролерот кој ќе се обработува во ова поглавје е од тип V&R (Bernecker & Rainer) модел X20 CP1485. Исто така ќе се опишат и неговите додатни модули: X20 DI9371 – дигитален влезен модул, X20 AI4622 – аналоген влезен модул, X20 DO 9322 – дигитален излезен модул, X20 AO 4622 – аналоген излезен модул како и операторскиот интерфејс, Touch Screen дисплеј, исто така производ на V&R, модел 4PP320.0571.35.

### 5.1. Карактеристики на X20 системите

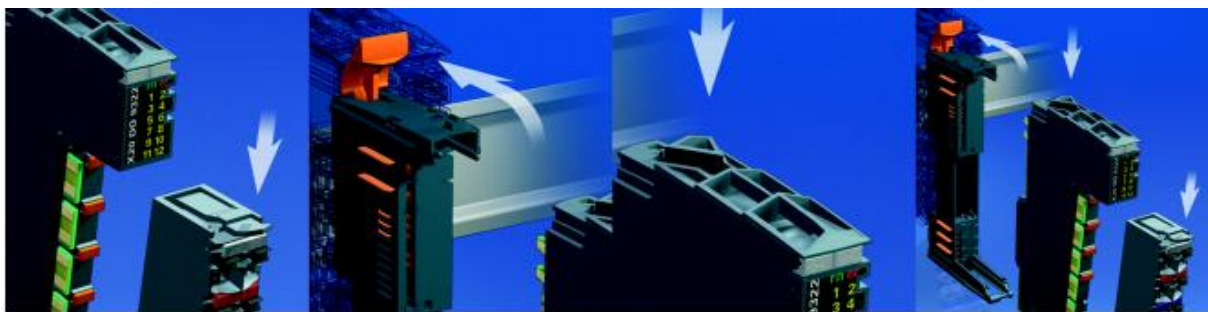
#### 5.1.1. Вовед

Со производството на X20 системите, компанијата V&R поставува нови стандарди согласни со нивното мото „Перфекција во автоматизацијата“. X20 системите претставуваат ново универзално решение за било каква задача од автоматизација на машини и производство. Тие се дизајнирани со помош на искуството стекнато од апликации од целиот свет, бројни разговори со корисници, со една единствена цел: попрост, поекономичен и побезбеден систем за практична примена.

Со своите добро смислени детали и софистициран ергономски дизајн, X20 системот е повеќе од далечински влезно/излезен систем. Тој претставува целосно решение за управување. Фамилијата на X20 системите овозможува да се комбинираат различни компоненти во зависност од барањата на корисникот и посебните барања во одредена апликација. Овој систем во комбинација со останатите компоненти (за некои од нив ќе стане збор подоцна) го достигнува максималниот потенцијал и овозможува имплементација на апликации со совршени перформанси и флексибилност.

Три основни елементи создаваат еден модул: приклучен блок, електронски блок и магистрален блок. Оваа модуларност резултира со систем што ги комбинира предностите на двата модула и влезно/излезните лизгачки модули:

- претходно поврзување без модулот
- приклучна електроника
- екстра магистрален слот за дополнителни опции



Слика 5.1 – Трите дела на X20 системот

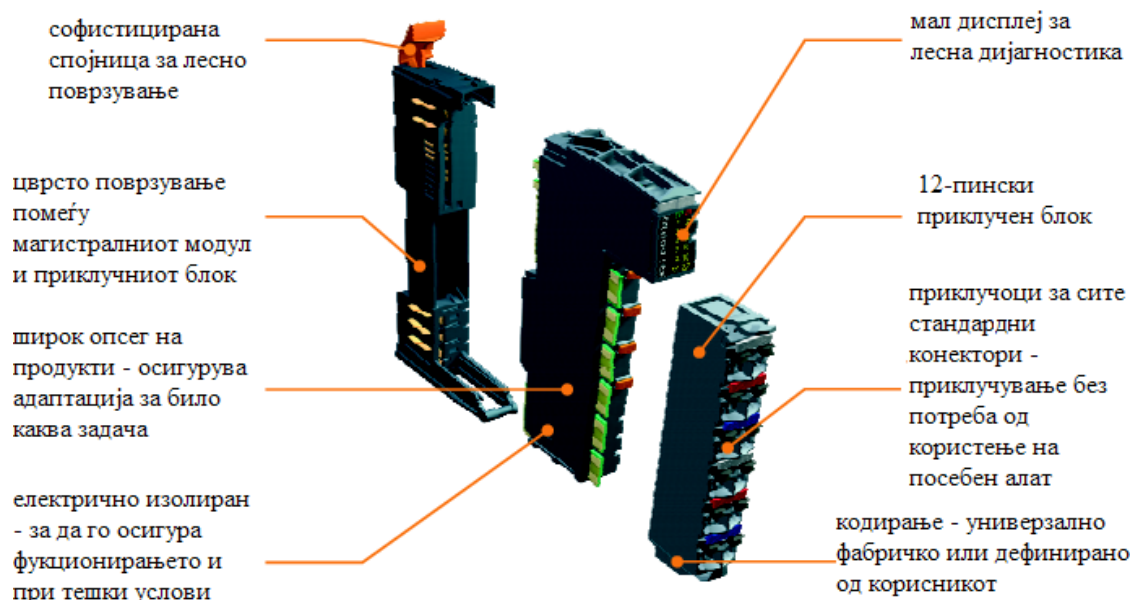


Системот X20 е за 50% подобрен во поглед на интегрираноста и просторната зафатнина на компонентите, усовершена технологија на поврзување и оптимална грануларност. Каналите (12 на број) се широки 12.5 mm овозможуваат одлична густина на компонентите и оптимална ергономија на приклучоците. Тој овозможува добра имплементација на едножично, двично и трижично поврзување без дополнителни џампери на приклучоците. Постои можност за максимална флексибилност, со едноканални и двоканални модули, така што се набавува само она што е потребно за апликацијата.

Термините модули, канали и сл. ќе бидат објаснети во продолжение.

Како што беше претходно споменато, X20 модулите се поделени на три дела, што овозможува многу едноставно искористување на можностите на контролерот во текот на целиот век на траење. Оваа поделба нуди многу можности:

- **Подесен за различни типови на машини.** Магистралните модули се основната платформа за различни типови на машини. Видот на машината, работата што таа ја врши и нејзиното функционирање одредува кои електронски модули ќе се применат. Софтверот го препознава хардверот што се користи и ги извршува потребните функции.
- **Конструкција со индустриски приклучни кутии.** Приклучните блокови на X20 системот се одвоени од електронските модули, што овозможува комплетно преврзување на приклучните кутии. Идеално е за машини за сериско производство.
- **Едноставно одржување.** Модулите можат да се заменат многу едноставно. Електронските модули можат да се заменат без да се прекинува работата на контролерот. Поврзувањето останува непроменето благодарение на одделните приклучни блокови. Оваа предност заштедува време.



Слика 5.2 – Составни делови и особини на еден X20 систем

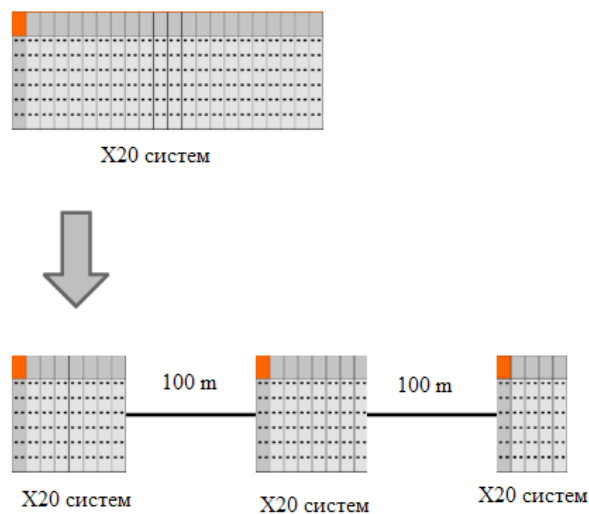
### 5.1.2. Централни процесорски единици (CPU) X20

Новиот, оптимално дизајниран асортиман на X20 процесорски единици задоволува широк опсег на употреби. Тие можат да се применат секаде, од стандардни апликации, до апликации со највисоки барања. Можат да манипулираат со времиња на циклусот од ред на 200  $\mu$ s.

Можностите за мрежно поврзување преку портовите RS232, Ethernet типот и USB веќе се стандардна опрема кај контролерите на V&R и нивното користење е возможно без никаква доплата. Освен тоа, секоја централно процесорска единица има и Powerlink конекција за комуникација во реално време (real-time communication). Овој тип на мрежно комуницирање е воведен од V&R и овозможува низ Powerlink мрежниот кабел покрај податоци да се пренесува и напојување за уредот поврзан во мрежата. Иако најголемиот дел од барањата ги исполнува и стандардната централна единица, X20 системите имаат и до три повеќенаменски слотови за дополнителни интерфејс модули.

Поради тоа што процесорските единици X20 се дизајнирани за монтирање на шина во метален орман, можат директно да се поврзат и до 250 влезно – излезни X20 модули, со 3000 канали. Ова овозможува највисоки перформанси, заедно со оддалечената (децентрализирана) матична плоча.

Децентрализираната матична плоча е местото каде можат да се поврзуваат разни модули (влезни, излезни, процесорски итн.) со помош X2X кабел. Тие можат да се постават на растојание и до 100 метри оддалеченост од главниот метален орман. Ова не претставува само обична плоча која е задолжена за комуникација помеѓу магистралните модули со помош на X2X кабел, туку овозможува пренос на податоците без користење на конвертори и без намалување на квалитетот на сигналот. Друга предност е фактот што напојувањето во централната процесорска единица интегрирано со влезните и излезните приклучоци за напојувањето, ги напојуваат оддалечената матична плоча, заедно со сензорите и актуаторите, така што не се потребни дополнителни компоненти. Со директно поврзување со централната процесорска единица, преку оддалечената матична плоча, може повеќекратно да се поврзуваат линии на влезови и излези било каде во опсег на 100 m.



Слика 5.3 – Поврзување на разни X20 модули со X2X кабел

Технологијата на изработка на овие управувачи единици е базирана на технологијата на последниот Intel Celeron процесор. Тие поседуваат голема RAM меморија, што на корисникот му овозможува неограничена слобода за разни видови на апликации. Системот е дополнет со SRAM меморија (статична RAM меморија), непрекинато напојувана од батерија, во која се складираат специфични податоци и заостанати променливи (варијабли). Во случај на прекин на напојувањето, променливите што се декларирани како заостанати автоматски се копираат од RAM меморијата во безбедната SRAM меморија. Податоците остануваат во такт се додека контролерот не се ресетира, и процесот продолжува да тече без никакви последици. Исто така, X20 системите се опремени и со слот за CompactFlash картички за зачувување на одредени податоци.

Овој систем е погоден за индустриски примени. Пружајќи највисоки перформанси, со многу стандардни интерфејси и интерфејси на надоградени модули, тој има и доста компактни димензии. Димензиите на централната процесорска единица (X20 CPU) и на X20 модулите се складни, со што максимално се искористува просторот во металниот орман каде што е сместен контролерот. Друга предност е фактот што не е потребен вентилатор за ниеден вид на процесори од X20 системот, освен за процесорите Celeron 650, застапени кај централните единици X20 1486 и X20 3486. Тоа овозможува речиси да не е потребно никакво одржување. За овие две процесора е предвиден еден заменлив вентилатор, и тоа без потреба од посебен алат, се заменува од надвор и при замената нема потреба да се вади процесорот (слика 5.4).

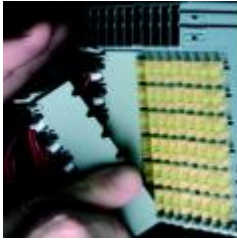


Слика 5.4 – Замена на вентилаторот кај Celeron 650 процесорите

### 5.1.3. Поврзување со приклучоците

Конструкцијата на индустрискиот метален орман ефикасно ги организира циклусите на производството. X20 системот поддржува ефикасно поврзување со користење на одвоени приклучни блокови. Целосната конфигурација на X20 системот е монтирана во металниот орман и поврзана со каблите. При тоа, не се потребни посебни кабли за дистрибуција на електрична енергија помеѓу X20 модулите и сензорите и актуаторите, со што поврзувањето се сведува на минимум.

Поврзувањето е брзо и не бара било каков посебен алат. Приклучните блокови имаат интегрирани конектори, при што жицата се поврзува со втиснување во приклучокот. Исто така, во секој приклучок може да се приклучи и наглавок со две жици, со дијаметар до  $2 \times 0,75 \text{ mm}^2$ . Поврзаните жици можат да се одврзат со шрафцигер. Исто така, на секој приклучок (терминал) има пристап за мерење на напонот на приклучокот.



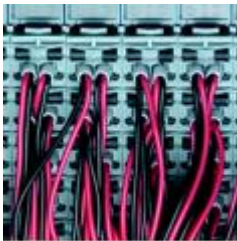
Приклучоците претходно можат да се поврзат одвоено од вистинските влезно – излезни модули. Ова е голема предност во поврзувањето, што обезбедува одвоено производство, навремена логистика и инсталација на претходно склопени системи.



Едноставно и брзо поврзување без користење од посебен алат. Жицата се поврзува со втиснување во приклучокот. Достапни се приклучоци со 6 и 12 пинови.



Со фабричкото кодирање се спречува поврзување на несоодветни и опасни делови на системот. Со ова се гарантира дека можат да се поврзат само оние делови што можат да се комбинираат.



Густината на компонентите не мора да биде на штета на ергономијата. Приклучоците се одделени околу 5mm, со што се обезбедува добра прегледност.



Со кодирање во апликацијата, се спречува неправилно вметнување на приклучоците. Ако тие неправилно се поврзат, електрониката можеби нема да се оштети, но системот нема да функционира.

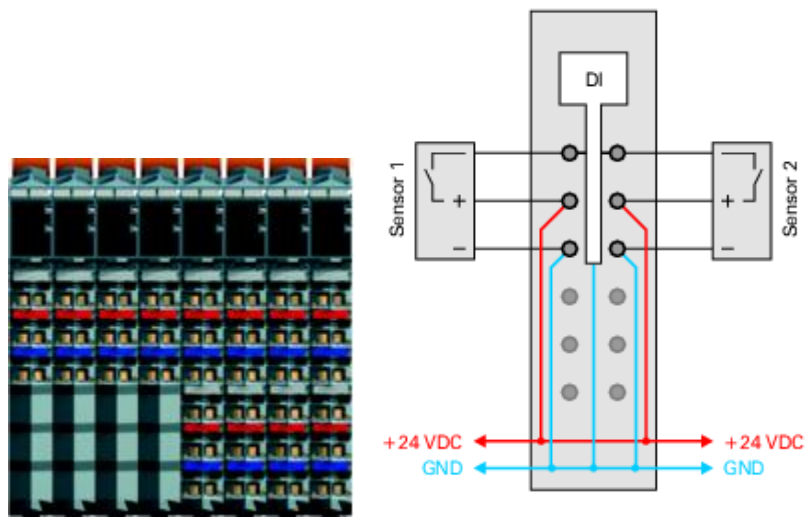


Секој приклучок е јасно означен на пластиката, со што се знае кој приклучок од кој сензор доаѓа или кон кој актуатор води.

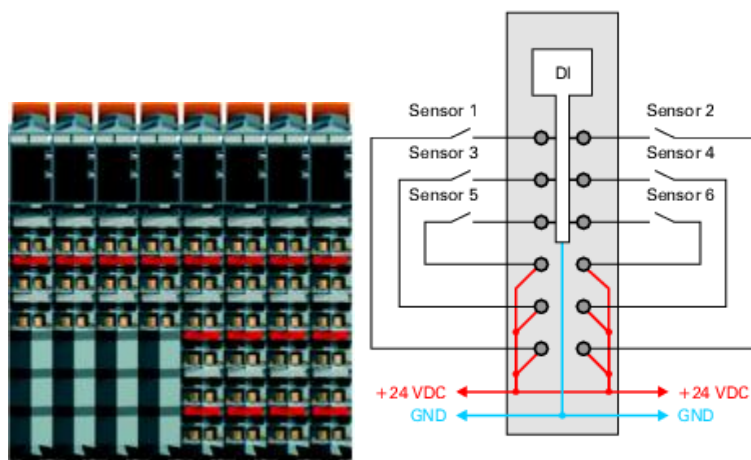


Како додаток на приклучниот конектор и механизмот за отклучување, секој приклучок овозможува пристап за испитување на напонот на приклучокот, без да се одврзува жицата.

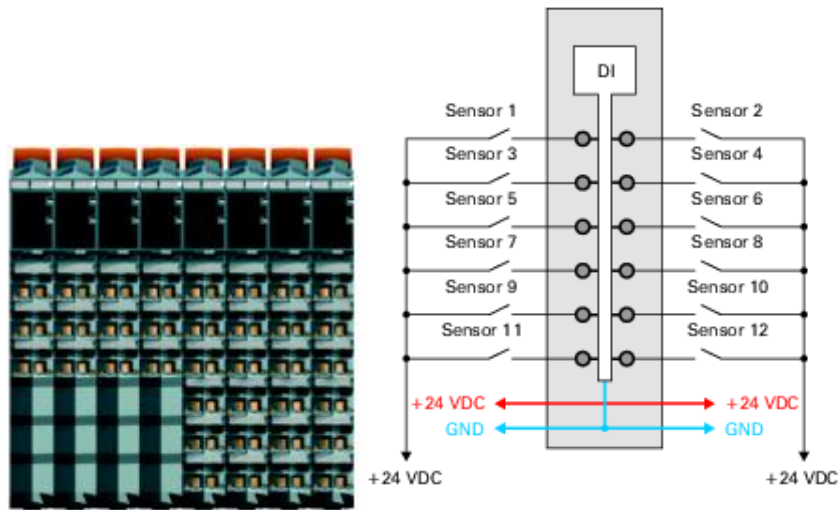
Поврзувањето може да се изведе едножично, двојично или трижично. При тоа не се потребни дополнителни џампери и сите три вида на поврзување може да се комбинираат.



Слика 5.5а – Класично трижично поврзување (интегрирано напојување и заземјување со сензорите и актуаторите)



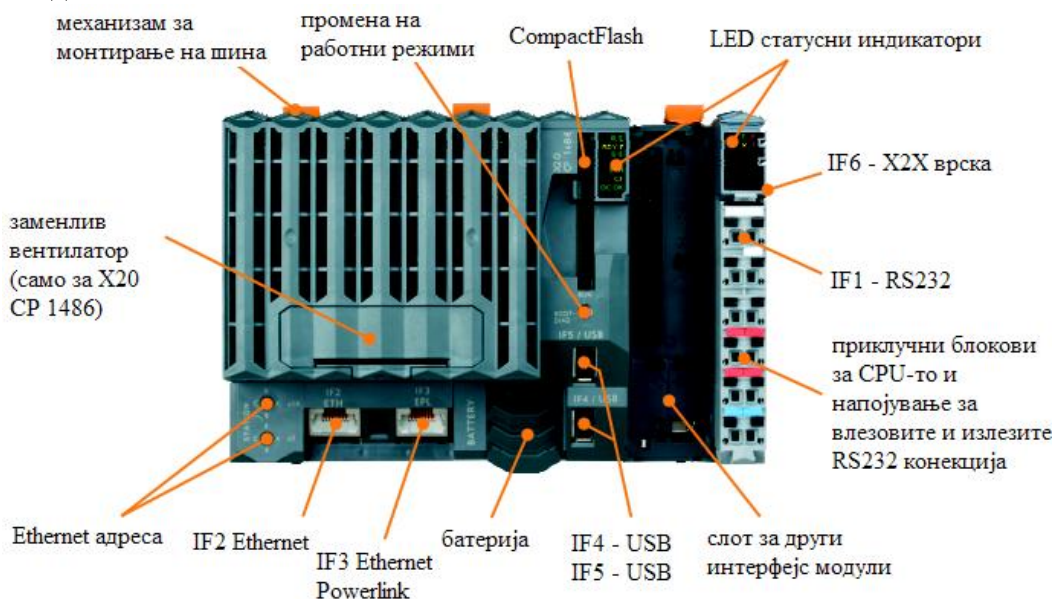
Слика 5.5б – Класично двојично поврзување (не се потребни дополнителни приклучоци)



Слика 5.5в – Класично едножично поврзување


## 5.2 Опис на контролерот тип X20 CP1485

Програмибилниот логички контролер од типот X20 CP1485 е базиран на процесорската технологија на Intel Celeron 400 и може да задоволи широк спектар на барања. Располага со 32 MB DRAM (динамички RAM) и 1 MB SRAM (статички RAM) мемории, а исто така и со преносна меморија CompactFlash. Самата процесорска единица поседува разни видови на интерфејси за комуникација со други уреди, како персонален компјутер, други програмибилни контролери итн. Тоа се: еден слот за X20IF модули, два USB интерфејса, еден RS232 интерфејс, еден Ethernet мрежен интерфејс, еден Ethernet Powerlink мрежен интерфејс. На следната слика 5.6 е прикажан контролерот X20 CP1485 со составните делови.



Слика 5.6 – Составни делови на X20 CP1485 контролерот (важи и за типовите X20 CP1484 и X20 CP1486.)

За поедноставна дијагностика на контролерот се наоѓаат LED статусни индикатори, и тоа на две места: едниот е за процесорот, а другиот за внатрешното напојување. Подолу се објаснети двата индикатори.

<b>Статусни индикатори за процесорот</b>			
			
<b>LED индикатор</b>	<b>Боја на светлото</b>	<b>Статус</b>	<b>Опис</b>
R/E	зелена	вклучено	Програмата се извршува.
	црвена	вклучено	Сервисирање.
RDY/F	жолта	вклучено	Процесорот е активен.
	црвена	вклучено	Превисока температура.
S/E	зелена/ црвена		Сигнал за статус и неисправност.
EPL	зелена	вклучено	Powerlink врската е воспоставена.
		трепка	Powerlink врската е воспоставена и светлото трепка при пренос на податоци низ магистралата.
ETH	зелена	вклучено	Воспоставена е врска на Ethernet мрежата.
		трепка	Воспоставена е врска на Ethernet мрежата и светлото трепка при пренос на податоци низ магистралата.
CF	жолта	вклучено	CompactFlash–от е во ред.
	зелена	вклучено	CompactFlash–от е активен.
DC OK	зелена	вклучено	Напојувањето на процесорот е во ред.
	црвена	вклучено	Батеријата е празна.

Индикаторот за статус и неисправност е зелен или црвен. Тој има различни значења за различни режими на работа, а тие значења највеќе се однесуваат за видот на неисправноста.

## Статусни индикатори за внатрешното напојување



LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Неповрзано напојување на модулот.
		трепка еднаш	Ресетирање.
		трепка	Подготвителен режим на работа.
		вклучено	Работен режим (вклучено).
e	црвено	исклучено	Неповрзано напојување на модулот или се е во ред.
		трепка двапати	Означува еден од следните случаи: - Преоптоварено напојување на X2X врската - Напојувањето на влезовите и излезите е премногу ниско - Влезниот напон на X2X врската е премногу низок
e + r	непрекинато црвено / зеленото трепка еднаш		неисправен фирмвер (firmware – вграден софтвер во хардвер кој не може да се модифицира)
S	жолто	исклучено	Нема проток на информации низ RS232 интерфејсот
		вклучено	Се испраќаат или примаат податоци низ RS232 интерфејсот
I	црвено	исклучено	Вредноста на напојувањето на X2X врската е во нормални граници
		вклучено	Преоптоварено напојување на напојувањето на X2X врската

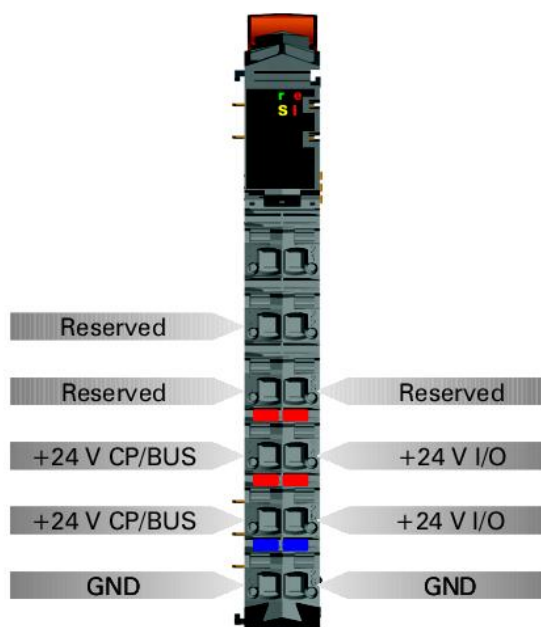
За да работи процесорот, потребна е меморија за програмата. Програмата се складира во CompactFlash меморијата. Овој тип на меморија не е стандарден при набавката на контролерот и потребно е посебно да се набави. Важна работа на која треба да се внимава е CompactFlash картичката да не се вади од слотот за време на работата на контролерот.

Со посебен прекинувач може да се променат три режими на работа на контролерот.

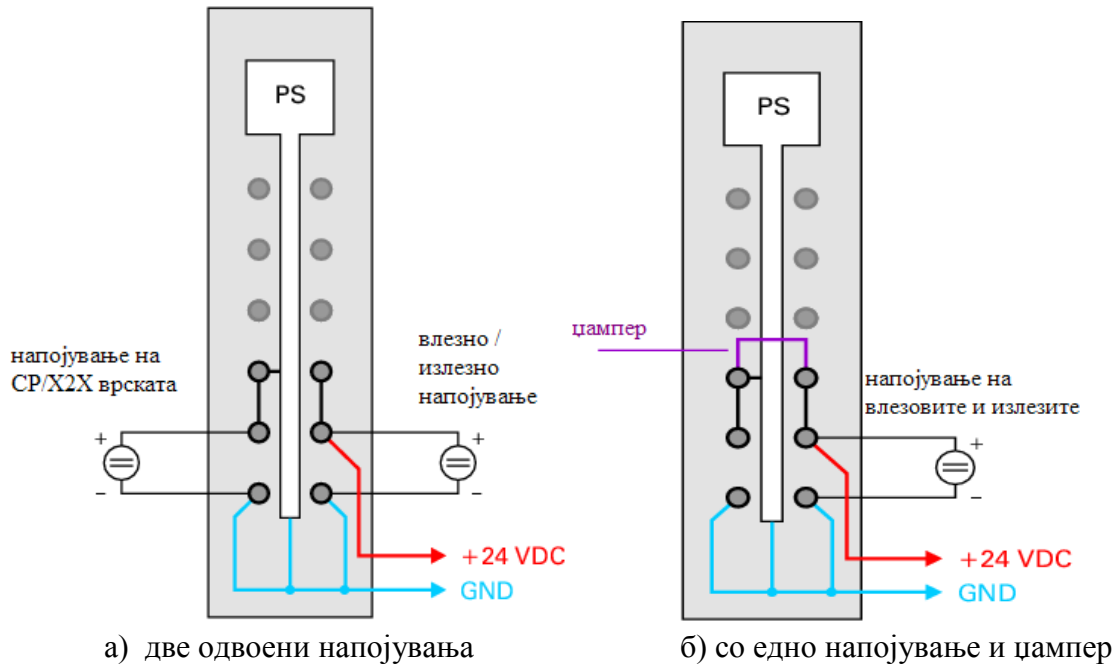


Прекинувач за промена на режимите на работа		
		
Позиција на прекинувачот	Работен режим	Опис
BOOT	Бутирање	Во оваа позиција на прекинувачот се стартува стандардниот B&R Automation Runtime (AR) и системот за извршување може да се инсталира користејќи поврзан (online) интерфејс (B&R Automation Studio).
RUN	Извршување	Режим на извршување
DIAG	Диагностика	Процесорот бутира во работен режим дијагностика. Програмските сегменти во RAM-от и FlashPROM-от не се иницијализирани. Кога ќе завши овој работен режим, процесорот бутира со т.н. топол рестраст (софтверско ресетирање на процесорот).

Напојувањето на процесорот доаѓа како интегрирано со X20 процесорите. Опремено е со напојување на процесорот, X2X врквата и внатрешните влезови и излези. Напојувањето за процесорот и X2X врквата е електрично изолирано.



Слика 5.7 – Значење на пиновите на интегрираното напојување

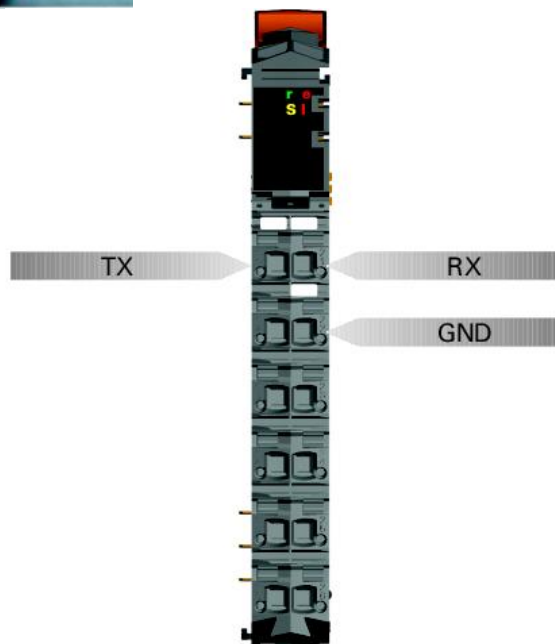


Слика 5.8 – Примери на поврзување со напојувањето

Интерфејсот RS232 не е електрично изолиран. Тој се користи како online интерфејс за комуникација со уредот за програмирање (слика 5.9).



Ознаката IF2 го означува Ethernet интерфејсот за локални мрежи.



Слика 5.9 – Значење на пиновите на RS232 интерфејсот (IF1)

### 5.3. Дигитален влезен модул X20 DI9371

Дигиталниот влезен модул X20 DI9371 (слика 5.10) е опремен со 12 влезови за едножично поврзување, од типот синк. Кај другите дигитални влезни модули наменети за X20 системот постојат варијации во смисла на типот на поврзување (влезни модули наменети за двојично и трижично поврзување), типот на поврзување (синкинг или сурсинг) и бројот на влезови (шест за двојично и три за трижично поврзување – слика 5.5). Во конкретна апликација, со изборот на типот на поврзување на дигиталните влезови од сензорите, потребно е да се избере соодветен дигитален влезен модул. За конкретниот модул X20 DI9371 потребни се и магистрален модул (Bus module) X20 BM11 и приклучен блок (Terminal block) X20 TB12 (слика 5.10).

Влезниот напон на дигиталните влезови е 24 V ( $\pm 15 \div 20\%$ ). При влезен напон од 24 V, влезната струја е 3,75 mA, а влезната отпорност 6,4  $\Omega$ . Ниското напонско ниво е за вредности  $< 5$  V, а високото напонско ниво за вредности  $> 15$  V. Сигналите од дигиталните влезови (на пр. сензорите) можат да се обработуваат во контролерот филтрирани или нефилтрирани. Дигиталниот влезен модул нуди можност на филтрирање на сигналите. И кај нефилтрираните, и кај филтрираните сигнали, статусот на сигналот се регистрира со фиксирана компензација, со запазување на циклусот на мрежата и се пренесува во истиот циклус. Филтрирањето се извршува асинхроно со мрежата со доцнење од 200  $\mu$ s заедно со временско отстапување од 50  $\mu$ s поради преносот низ мрежата.

Во табелата под слика 5.10 се објаснети значењата на статусните индикатори на дисплејот.



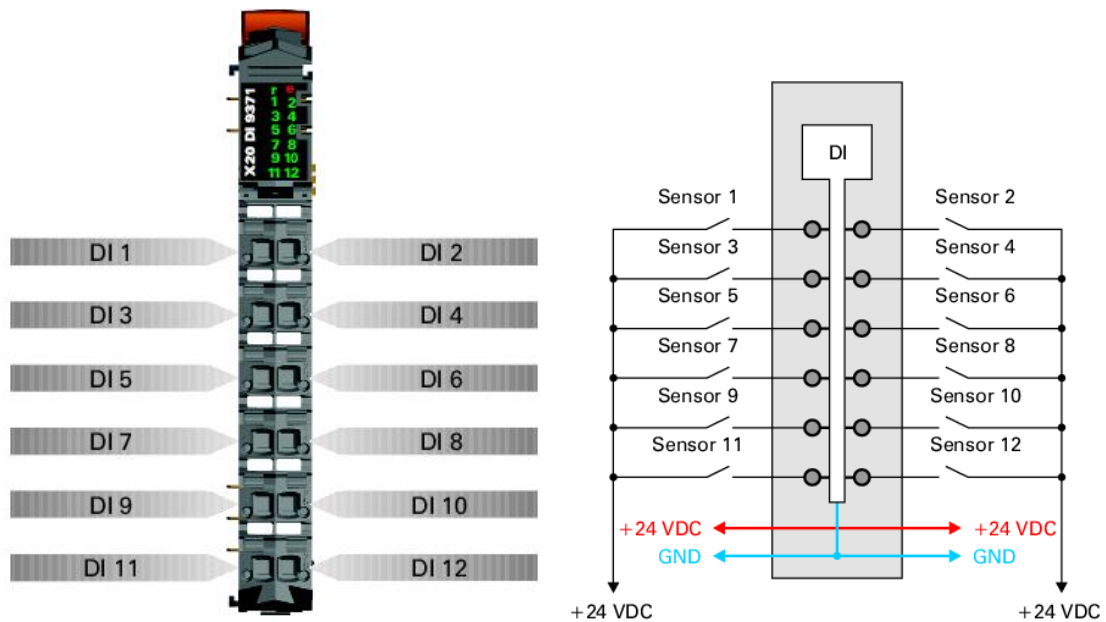
Слика 5.10 – Дигитален влезен модул X20 DI9371 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

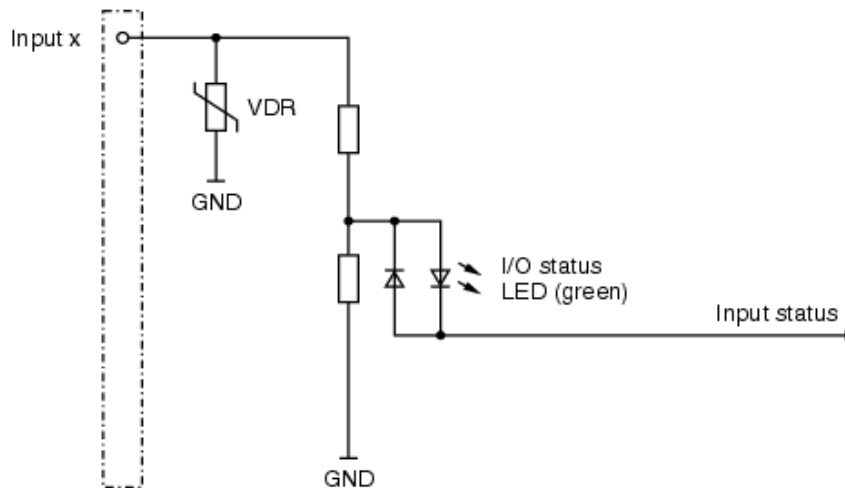
## Статусни индикатори на влезниот модул



LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Напојувањето на модулот не е поврзано.
		трепка еднаш	Ресетирање.
		трепка	Подготвителен режим на работа.
		вклучено	Режим на работа.
e	црвено	исклучено	Напојувањето на модулот не е поврзано или се е во ред.
e + r	непрекинато црвено / зеленото трепка еднаш		Неисправен фирмвер.
1 – 12	зелено		Статус на соодветниот дигитален влез.

На наредната слика 5.11 се прикажани значењата на пиновите, пример за едножичното поврзување, како и шема на влезното струјно коло.





Слика 5.11 – Значење на пиновите, пример за едножичното поврзување и шема на влезното струјно коло

Минимално време на циклусот е најмалото време на запирање на циклусот без да настане комуникациска грешка. Треба да се напомене дека кај брзите циклуси времето на мирување за мониторинг на управувањето, дијагностика и ациклични команди е намалено. Минималното време на циклусот за нефилтрирани сигнали изнесува  $\geq 100 \mu\text{s}$ , а за времиња на циклусот  $\leq 150 \mu\text{s}$ , филтрирањето се деактивира. Минималното време на ажурирање на влезите и излезите е во врска со минималното време на запирање на циклусот, така што во секој циклус се случува ажурирање на влезите и излезите. Тоа време за нефилтрирани сигнали е  $\geq 100 \mu\text{s}$ , а за филтрирани  $\geq 100 \mu\text{s}$ .

#### 5.4. Аналоген влезен модул X20 AI4622

Аналогниот влезен модул AI4622 е опремен со четири влеза со 12 битна дигитална резолуција на A/D конверзијата. Со користење на различни точки на приклучок, може да се регистриваат сигналите на струјата и напонот. За конкретниот модул X20 AI4622 потребни се и магистрален модул (Bus module) X20 BM11 и приклучен блок (Terminal block) X20 TB12 (слика 5.12).

Влезниот напон треба да биде  $\pm 10 \text{ V}$ , а максимално дозволеениот влезен напон на аналогните влезови е  $\pm 30 \text{ V}$ . Влезната струја може да биде во опсегот од 0 mA до 20 mA, или од 4 mA до 20 mA. Предност кај долната граница на вредноста на струјата од 4 mA во однос на онаа од 0 mA, е во редуцирањето на можноста сигналот да се изгуби поради појава на шум. Исто така, најниската вредност на сигнал од 4 mA може да се искористи како извор на енергија за сензори или друга опрема. Највисоката дозволена влезна струја е  $\pm 50 \text{ mA}$ . Времето на A/D конверзија е 300  $\mu\text{s}$  за сите влезови. Отпорноста на приклучениот влезен уред треба да биде помала од 400  $\Omega$ . По A/D конверзијата, 12 битната променливата е од тип INT (Integer). При тоа, најмалку важниот бит (LSB – Least Significant Bit) за напон е  $1 \text{ LSB} = \frac{8001}{2} = 2.441 \text{ mV}$ , а за струја  $1 \text{ LSB} = \frac{8008}{2} = 4.883 \mu\text{A}$ .

Во табелата под слика 5.12 се објаснети значењата на статусните индикатори на дисплејот.



Слика 5.12 – Аналоген влезен модул X20 AI4622 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

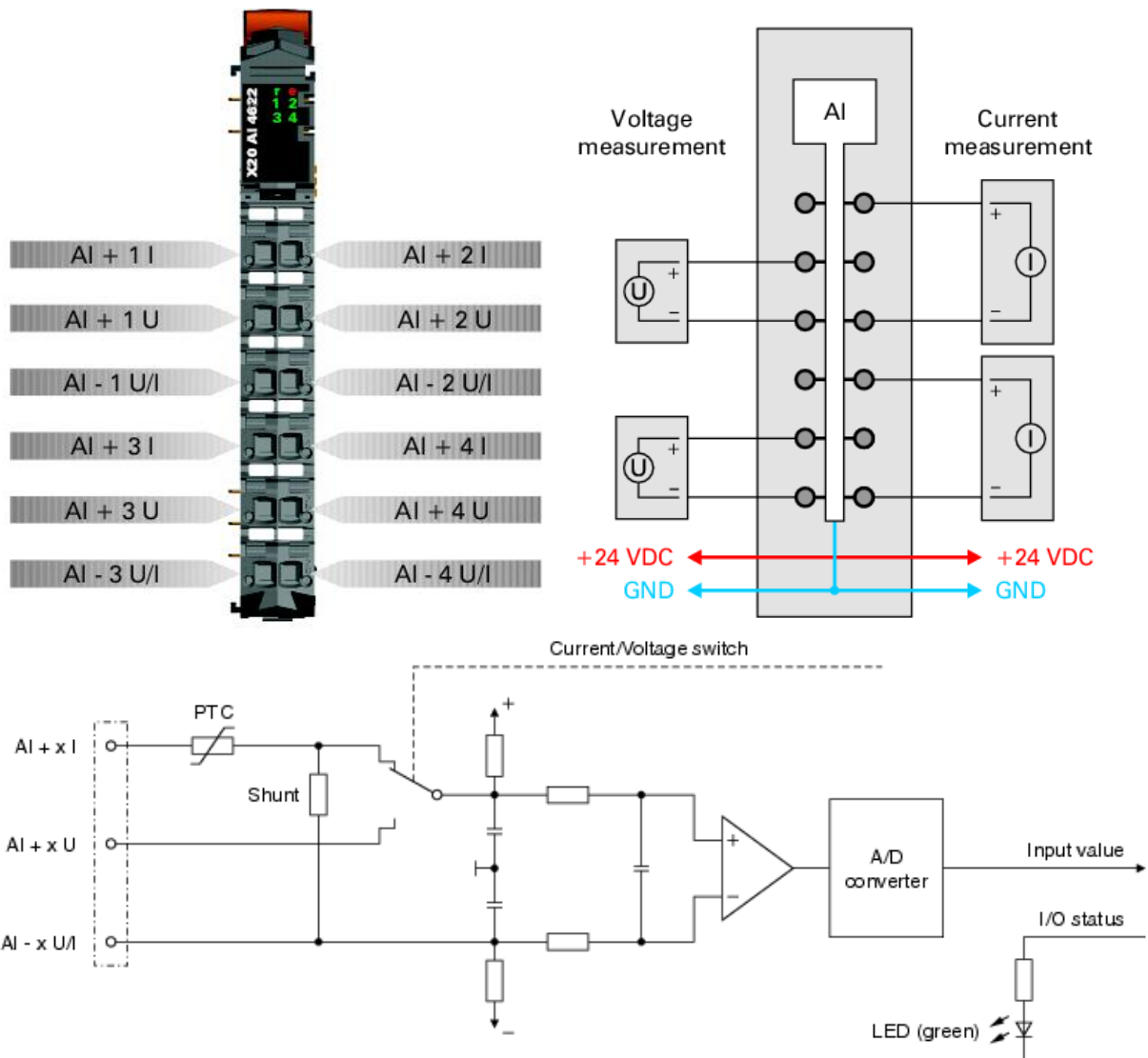
### Статусни индикатори на влезниот модул



LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Напојувањето на модулот не е поврзано.
		трепка еднаш	Ресетирање
		трепка	Подготвителен режим на работа
		вклучено	Режим на работа
e	црвено	исклучено	Напојувањето на модулот не е поврзано или се е во ред.
		вклучено	Неисправна состојба или ресетирање
e + r	непрекинато црвено / зеленото трепка еднаш		неисправен фирмвер

1 – 4	зелено	исклучено	Отворена конекција или сензорот е неповрзан
		трепка	Вредноста на влезниот сигнал е превисока или прениска
		вклучено	A/D конверторот работи, вредноста е во ред

На наредната слика се прикажани значењата на пиновите, пример за поврзување и е дадена шема на влезното струјно коло.



Слика 5.13 – Значење на пиновите, пример на поврзување со аналогни влезови и шема на влезното коло

Овој модул е опремен со подесив влезен филтер на сигналте. Минималното време на циклусот мора да биде поголемо од 500µs. За пократки времиња на циклусот филтерот

е деактивиран. Ако влезниот филтер е активен, тогаш каналите се скенираат во различни циклуси, со временска разлика од 200  $\mu$ s. A/D конверзијата е асинхрона на циклусот на мрежата.

Сите канали (влезови) се опремени за струјни или напонски влезни сигнали (слика 5.13). Видот на сигналот е определен од приклучоците кои се искористени (на кои има приклучено влез). Бидејќи за струјните и за напонските сигнали се потребни различни подесувања, потребно е да се подеси саканиот тип на влезниот сигнал:

- $\pm 10$  V за напонски сигнал (стандардна вредност)
- од 0 до 20 mA за струен сигнал
- од 4 до 20 mA за струен сигнал

Влезниот сигнал се контролира преку горната и долната гранична вредност. Стандардните гранични вредности се дадени во табелата:

Стандардна гранична вредност	Напонски сигнал $\pm 10$ V		Струен сигнал (0 до 20 mA)		Струен сигнал (4 до 20 mA)	
	Горна граница	+ 10V	+32767(\$7FFF)	20 mA	+32767(\$7FFF)	20 mA
Долна граница	- 10V	-32767(\$8001)	0 mA	0	4 mA	0

Други гранични вредности се дефинираат ако е потребно. Граничните вредности важат за сите влезови (каналите). Тие се активираат со внесување на вредноста на границите во регистарот. Откако ќе се постават нови граници, аналогните сигнали се контролираат според новите граници.

Во случај на влезен струен сигнал, подесен за вредности 4 mA до 20 mA, за да се измерат струи помали од 4 mA, мора да се постави негативна гранична вредност. Така, 0 mA ќе биде еднаква на -6553 (\$E667). За да се ограничи најниската вредност на напонот на 0 V, потребно е наместо -32767(\$8001), да се внесе 0. Истото важи и за струјните сигнали.

Минималното време на циклусот е минималното време потребно за запирање на циклусот на пренос на информациите без да настане комуникациска грешка. Брзите циклуси имаат намалено време на мирување, потребно за манипулација со мониторингот, дијагностиката и ацикличните команди. За нефилтрирани влезови тоа време е поголемо или еднакво на 100  $\mu$ s, а за филтрирани влезови поголемо од 500  $\mu$ s.

Минималното време за ажурирање на влезовите и излезите се однесува на минималното време потребно за запирање на циклусот на пренос на информациите, така што во секој циклус се случува ажурирање на влезовите и излезите. За нефилтрирани влезови тоа време е 300  $\mu$ s (за сите влезови), а за филтрирани влезови  $\geq 1$  ms.



## 5.5. Дигитален излезен модул X20 DO9322

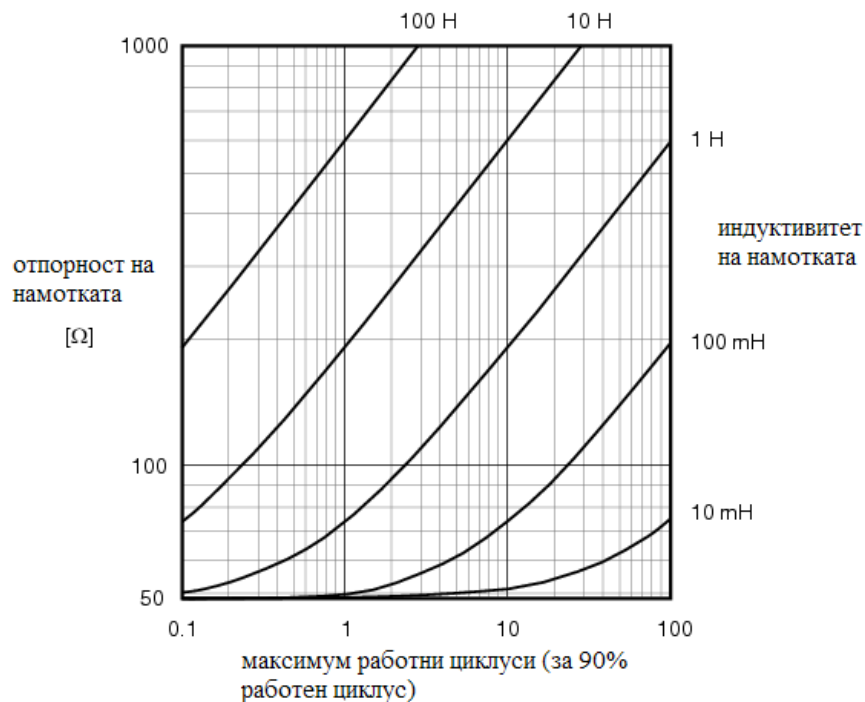
Дигиталниот излезен модул X20 DO9322 е опремен со 12 дигитални излези за едножично поврзување наменети за сурс (Source) поврзување со излезите. Кај другите дигитални излезни модули наменети за X20 системот постојат варијации во смисла на типот на поврзување (излезни модули наменети за двојично и трижично поврзување), типот на поврзување (синк или сурс) и бројот на излези (на пример, шест за двојично и четири за трижично поврзување). Во конкретна апликација, со изборот на типот на поврзување на дигиталните излези со актуаторите, потребно е да се избере соодветен дигитален излезен модул. За конкретниот модул X20 DO9322 потребни се и магистрален модул (Bus module) X20 BM11 и приклучен блок (Terminal block) X20 TB12 (слика 5.14).



Слика 5.14 – Дигитален излезен модул X20 DO9322 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

Номиналниот излезен напон е 24 V, а номиналната излезна струја 0.5 A. Излезите се заштитени од прекумерна струја или краток спој. Преминот на логичките вредности на напонот се изведува со FET транзистори. Мониторингот на излезите се извршува со паузи од 10 ms. При исклучен излез постои таканаречената струја на истекување (leakage current) и таа изнесува 5 $\mu$ A. Максималната струја при краток спој е помала од 12 A. Времето на вклучување после прекин поради краток спој или преоптоварување е околу 10 ms (зависи од температурата на модулот). Времето на задржување при премин од 0 во 1, како и од 1 во 0 е помало од 300  $\mu$ s. Максималната фреквенција на вклучување и исклучување за отпорнички излези е 500 Hz. За индуктивни излези (со 90 % работен циклус) фреквенцијата на вклучување и исклучување може да се види од дијаграмот на слика 13. Запирниот напон при исклучување на индуктивни излези е 50 VDC.

Во табелата под слика 5.15 се објаснети значењата на статусните индикатори на дисплејот.



Слика 5.15 – Вклучување и исклучување на индуктивни излези

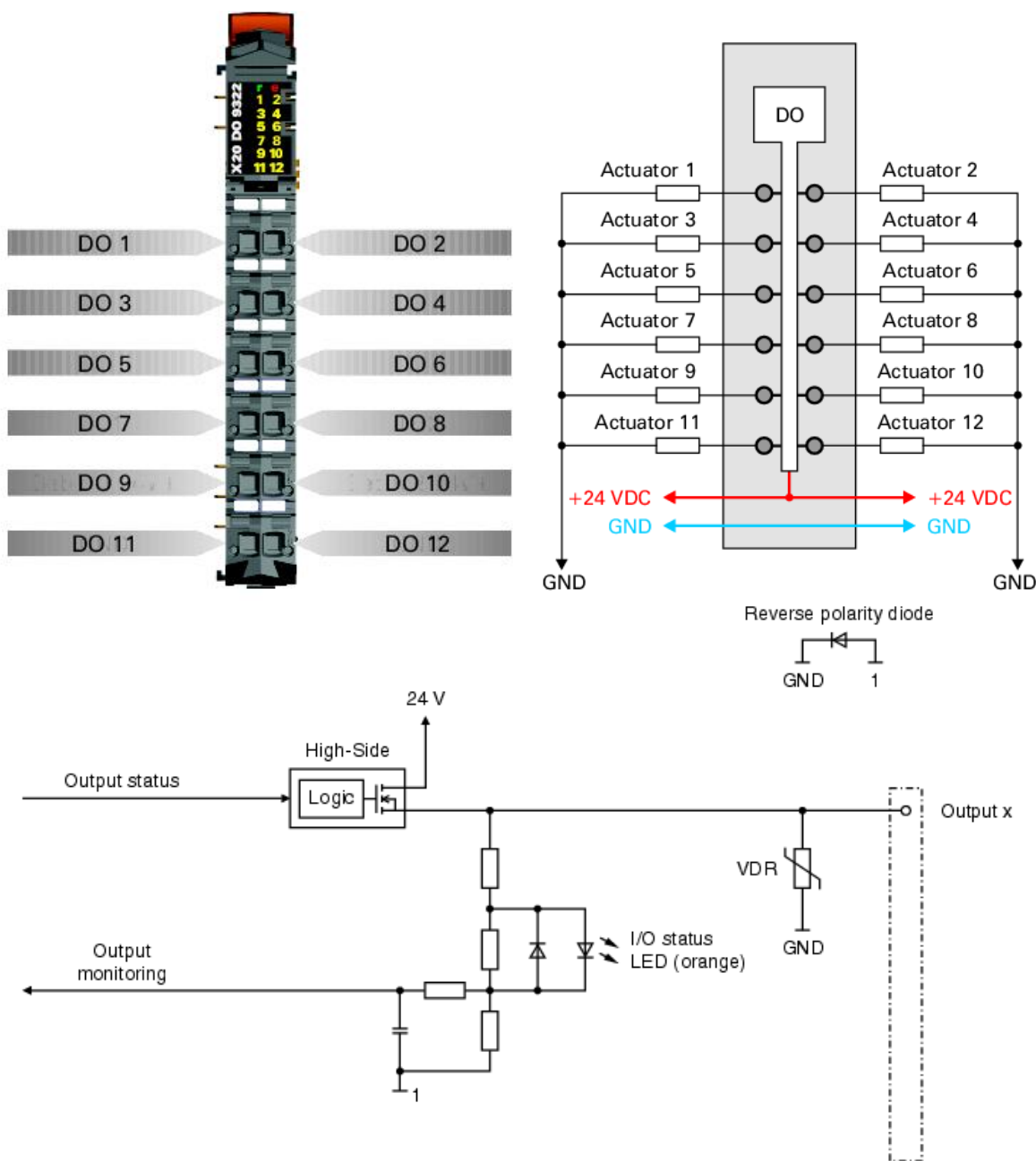
### Статусни индикатори на излезниот модул



LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Напојувањето на модулот не е поврзано.
		трепка еднаш	Ресетирање
		трепка	Подготвителен режим на работа
		вклучено	Режим на работа
e	црвено	исклучено	Напојувањето на модулот не е поврзано или се е во ред.
		трепка еднаш	Предупредување / грешка на некој влезен или излезен канал. Активиран е мониторинг на нивоата на дигиталните излези

e + r	непрекинато црвено / зеленото трепка еднаш	неисправен фирмвер
1 – 12	портокалово	Статус на соодветниот дигитален излез.

На наредната слика се прикажани значењата на пиновите, пример за поврзување и е дадена шема на излезното струјно коло.



Слика 5.16 – Значење на пиновите, пример на поврзување со дигитални излези и шема на излезното коло

Минималното време на циклусот е поголемо или еднакво на 100  $\mu$ s. Минималното време за ажурирање на влезовите и излезите е еднакво на минималното време на циклусот.

## 5.6. Аналоген излезен модул X20 AO4622

Аналогниот излезен модул X20 AO4622 е опремен со четири излези со 12 битна дигитална резолуција на D/A конверзијата. Со користење на различни точки на приклучок, може да се избере помеѓу сигналите на струјата и напонот. За конкретниот модул X20 AO4622 потребни се и магистрален модул (Bus module) X20 BM11 и приклучен блок (Terminal block) X20 TB12 (слика 5.17).



Слика 5.17 – Аналоген излезен модул X20 AO4622 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

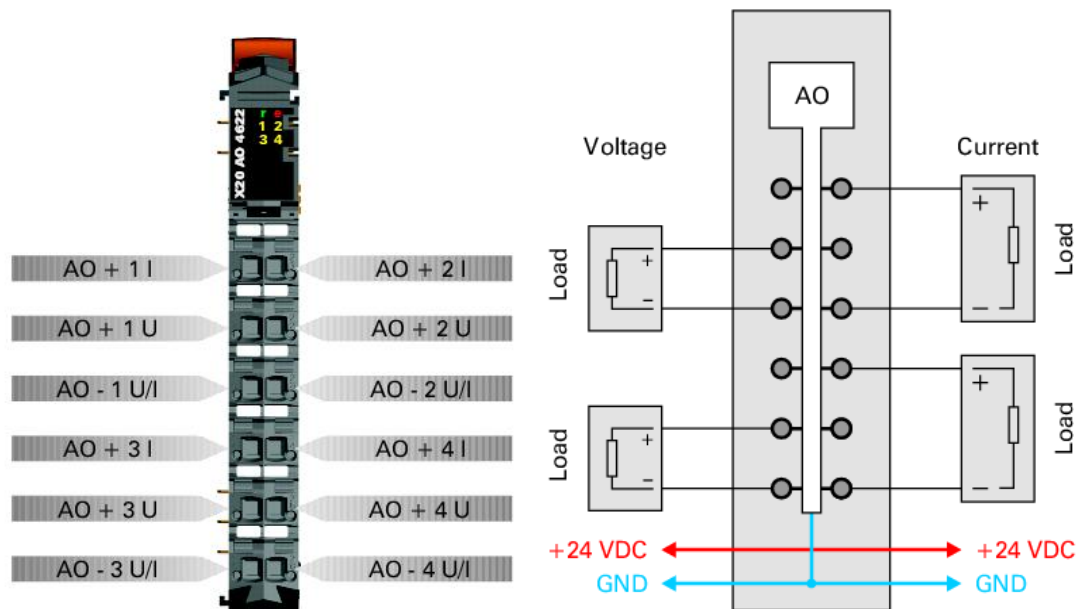
Излезниот напон е во границите од  $\pm 10$  V, а излезната струја во границите  $0 \div 20$  mA. Времето потребно за D/A конверзија е 300  $\mu$ s за сите излези. Излезите се заштитени од појава на краток спој со ограничување на струјата од  $\pm 40$  mA. Максимално дозволената отпорност на уредите приклучени на излезите е 600  $\Omega$ . Времето на одговор на промени на излезите во целиот опсег е 500  $\mu$ s. Овој модул поседува одредена нелинеарност на промена на параметрите, но таа е помала од 0,005 %, во зависност од опсегот на излезот. Пред D/A конверзијата, 12 битната променливата е од тип INT (Integer). При тоа, најмалку важниот бит (LSB – Least Significant Bit) за напон е  $1 \text{ LSB} = \frac{10}{2^{12}} = 4.882 \text{ mV}$ , а за струја  $1 \text{ LSB} = \frac{20}{2^{12}} = 9.766 \text{ }\mu\text{A}$ .

Во наредната табела се објаснети статусните индикатори на дисплејот. На сликата 5.18 под табелата се прикажани значењата на пиновите и пример за поврзување. На слика 5.19 е дадена шема на излезното струјно коло.

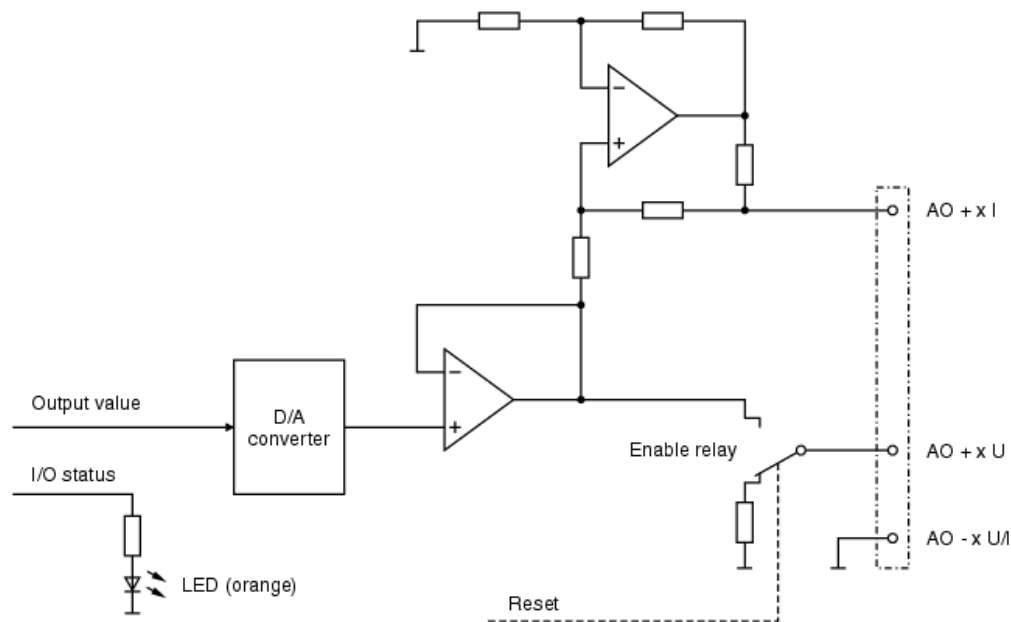
### Статусни индикатори на излезниот модул



LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Напојувањето на модулот не е поврзано.
		трепка еднаш	Ресетирање
		трепка вклучено	Подготвителен режим на работа. Режим на работа
e	црвено	исклучено	Напојувањето на модулот не е поврзано или се е во ред.
		вклучено	Неисправна состојба или ресетирање.
e + r	непрекинато црвено / зеленото трепка еднаш		неисправен фирмвер
1 – 4	портокалово	исклучено	Вредност = 0
		вклучено	Вредност ≠ 0



Слика 5.18 – Значење на пиновите и пример на поврзување со аналогни излези



Слика 5.19 – Шема на излезното струјно коло

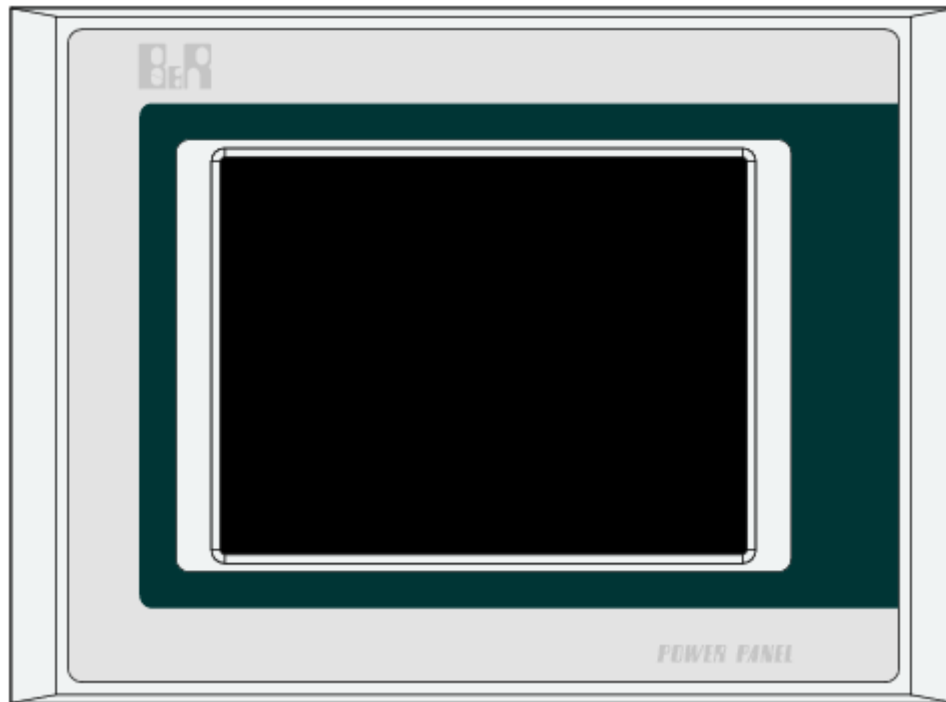
Секој канал (излез) може да се подеси за струјни или напонски сигнали. Видот на сигналот кој ќе се употреби за поврзување со актуаторите се одредува од приклучоците кои ќе се искористат (слика 5.18). Минималното време на циклусот е поголемо или еднакво на 250  $\mu$ s. Минималното време на ажурирање на влезовите и излезите што го дозволува овој модул, за сите излези е помало од 400  $\mu$ s.

## 5.7. Операторски интерфејс модел 4PP320.0571.35 (Touch Screen)

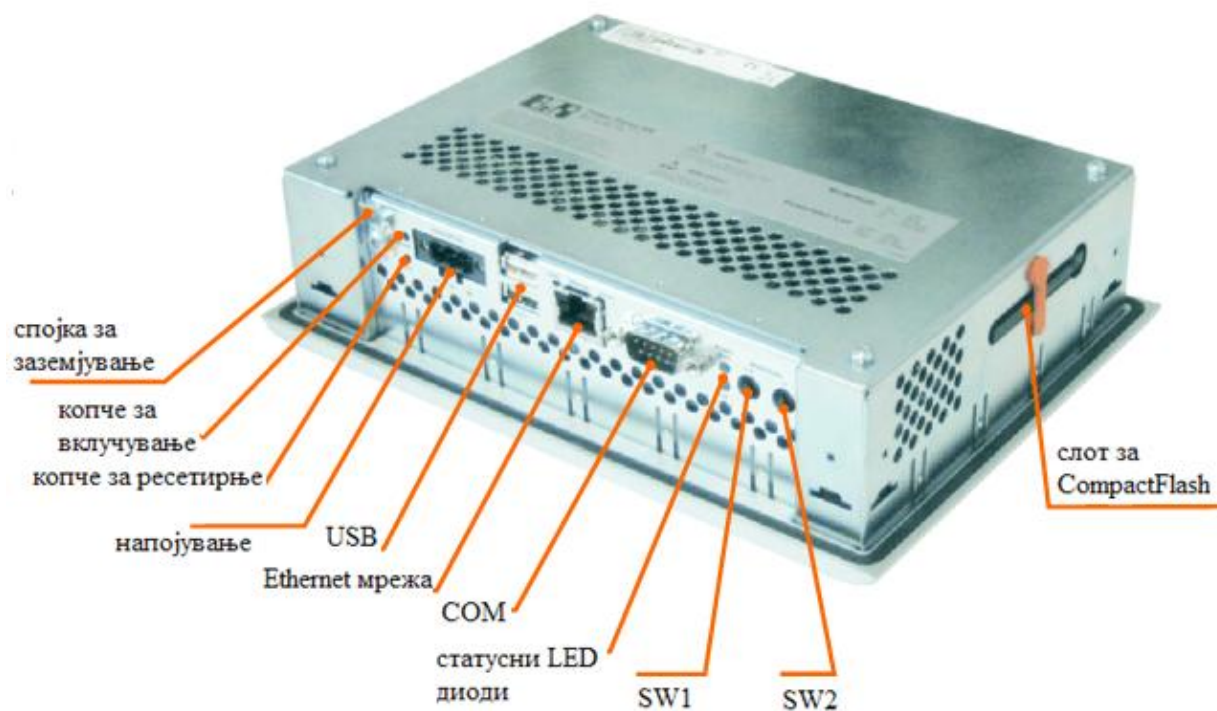
Операторскиот интерфејс (HMI – Human Machine Interface) служи за комуникација на операторот со машината или процесот. Тој нуди можност за внесување одредени параметри како влез во програмата, со што се овозможува одредени параметри од управувањето на процесот да ги задава операторот (пр. број на вртежи на некој мотор, насока на движење на некој подвижен дел итн.). Исто така, тој нуди можност и за прикажување на некои излезни параметри на дисплеј, кои ќе му дадат на операторот информација за состојбата на некој член од процесот (од актуатор или сензор), како на пример моментална брзина на некој мотор. Постојат разни видови на интерфејси, кои можат да се изберат во зависност од потребата на управувачкиот процес. Нашата лабораторија располага со Touch Screen интерфејс, модел 4PP320.0571.35 (слика 5.20 и 5.21), исто така производство на V&R. Тој ќе биде искористен за управувањето на процесот, разработен подолу.

Во наредната табела ќе бидат прикажани некои технички спецификации на интерфејсот.

Оперативен систем	Automation Runtime
<b>Процесор</b> L1 Cache – кеш меморија L2 Cache – кеш меморија Ладење	Geode LX800 500 MHz, 32-bit x86 128 KB (64 KB L cache / 64 KB D cache) 128 KB Пасивно
<b>Флеш меморија</b>	2 MB (за фирмверот)
<b>Меморија</b>	DDR SDRAM 128 MB
<b>Графика</b> Контролер Меморија	Geode LX800 8 MB заедничка меморија (резервирана на главната меморија)
<b>SRAM (Статичка RAM меморија)</b>	512 KB
<b>Батерија</b>	нема
<b>Ethernet мрежа</b> Контролер Брзина на пренос	Intel 82551ER 10/100 Mbps
<b>CompactFlash преносна меморија</b>	1 слот
<b>Сериски интерфејс</b> Брзина на пренос	RS232 Максимум 115 kBaud (kBaud - 1000 бита во секунда)
<b>USB интерфејс</b> Брзина на пренос	USB 1.1 и USB 2.0 до 480 Mbit/s
<b>Дисплеј</b> Резолуција	color TFT, 5,7 in (144 mm), 262 114 бои QVGA, 320 x 240
<b>Touch Screen</b> Контролер Степен на трансмисија	Аналоген, отпорнички Elo, 12 битен сериски До 80 % ± 5 %
<b>Електрични карактеристики</b> Номинален напон Номинална јачина на струја Потрошувачка на енергија Отпорност на заземјувањето	18 – 30 VDC 0,45 A околу 10 W 0 Ω
<b>Механички карактеристики</b> Надворешни димензии Метално кукиште Тежина <b>Работа при вибрации</b> при работа (непрекинати)  при работа (повремени) при чување при транспорт	ширина 212 mm, висина 156 mm, дебелина 55,5 mm  околу 1,4 kg  2 - 9 Hz: 1.75 mm амплитуда / 9 - 200 Hz: 0.5g 4.9 m/s <sup>2</sup>  2 - 9 Hz: 3 mm амплитуда / 9 - 200 Hz: 1g 9.8 m/s <sup>2</sup> 2 - 9 Hz: 7.5 mm, 9 - 200 Hz: 2 g, 200 - 500 Hz: 4 g 2 - 9 Hz: 7.5 mm, 8 - 200 Hz: 2 g, 200 - 500 Hz: 4 g



Слика 3.20 – Преден поглед на операторскиот интерфејс



Слика 3.21 – Заден поглед на операторскиот интерфејс

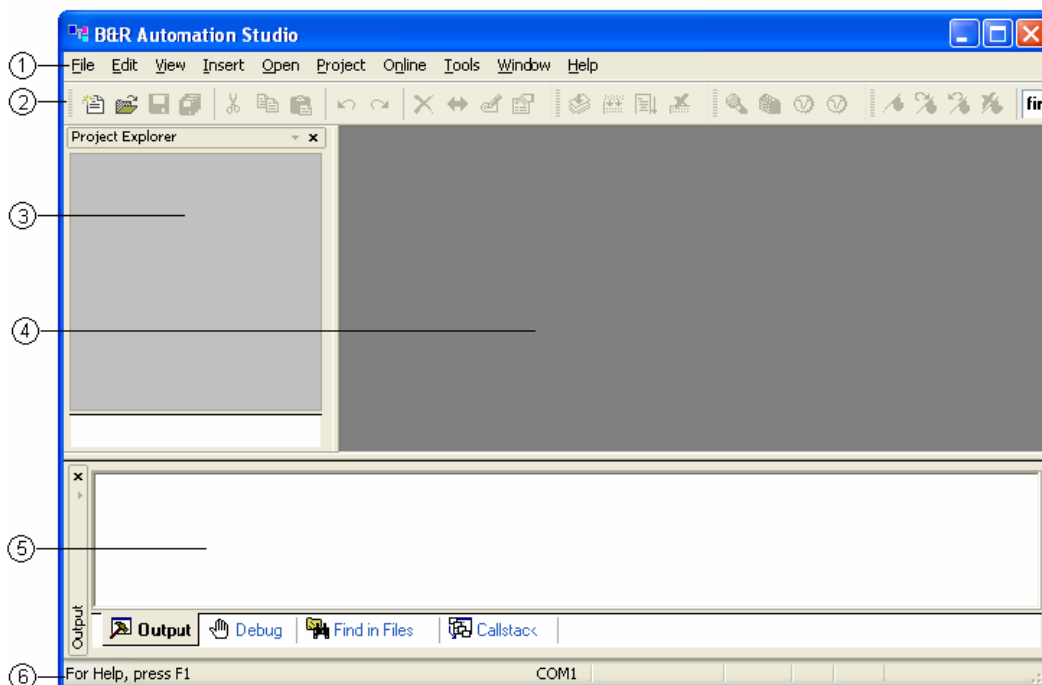


## 6. Анализа на програмскиот пакет од производителот V&R - Automation Studio 3.0

Програмскиот пакет Automation Studio претставува околина за програмирање на компонентите за автоматизација на V&R, што вклучува програмирање на контролерот, управување со движења и визуелизација (интерфејс). Неговата прегледна структура на проектите и можноста да оперира со широк спектар на различни конфигурации и варијации на машини значително го олеснува и помага процесот на програмирање. Покрај големиот број на алатки за дијагностика што ги поседува, на корисникот му се достапни различни програмски јазици (наведени подолу) и едитори за уредување на програмите. Со користење на стандардните библиотеки на V&R и IEC стандардите (International Electrotechnical Commission), кои се интегрирани во системот, се обезбедува висока ефикасност на текот на дејствата.

### 6.1. Основен прозорец на Automation Studio 3.0

Програмскиот пакет Automation Studio 3.0 се стартува како и секоја друга апликација инсталирана на компјутерот. При стартување на апликацијата се појавува прозорец како на слика 6.1.



Слика 6.1 – Основен прозорец на Automation Studio 3.0

Како што е прикажано на сликата, на прозорецот можат да се издвојат неколку битни области, означени со броеви. Тие се:

1. Главно мени (Main menu). Главното мени во V&R Automation Studio обезбедува пристап до сите можни функции што ги нуди програмскиот пакет.

2. Линија со алатки (Toolbar). Линијата со алатки содржи икони со кои е овозможен брз пристап до широк спектар на команди и функции. Линијата со алатки може да се уреди во мениото **View / Toolbars**.
3. Project Explorer. Кога проектот е отворен, во овој простор можат да се прикажат различни својства на проектот, кои можат да се задаваат, менуваат итн. Тие можат да се изберат со кликување на трите различни јазичиња (tabs): Logical view, Configuration view и Physical view.
4. Работен простор (Workspace). Ова е прозорецот каде што се прикажани фајлови проектот кој моментално е отворен.
5. Излезен прозорец (Output Window). Во овој прозорец се прикажуваат пораки од компајлерот (преведувачот на програмата), дебагерот (програмата за пронаоѓање на грешки) итн. На ова место се прикажуваат и резултатите од пребарувањето на функцијата “Find in Files”.
6. Статусна линија (Status bar). Статусната линија се наоѓа на дното од прозорецот и ги прикажува следните информации:
  - Кратка помош за командите од менијата или од линијата со алатки;
  - Кратки информации кои се однесуваат на процедурите на уредување;
  - Статусот на online врската помеѓу уредот за програмирање (компјутерот) и целниот уред;
  - Статусни податоци за моментално активниот прозорец.

Подетални информации за уредувачите (едиторите) и како тие се користат се дадени во следните поглавја.

## 6.2. Креирање на проект

Во ова поглавје детално ќе бидат разработени следните процедури:

- Креирање на проект
- Креирање на програма
- Компајлирање на проектот
- Трансфер на проектот
- Тестирање на програмската секвенца

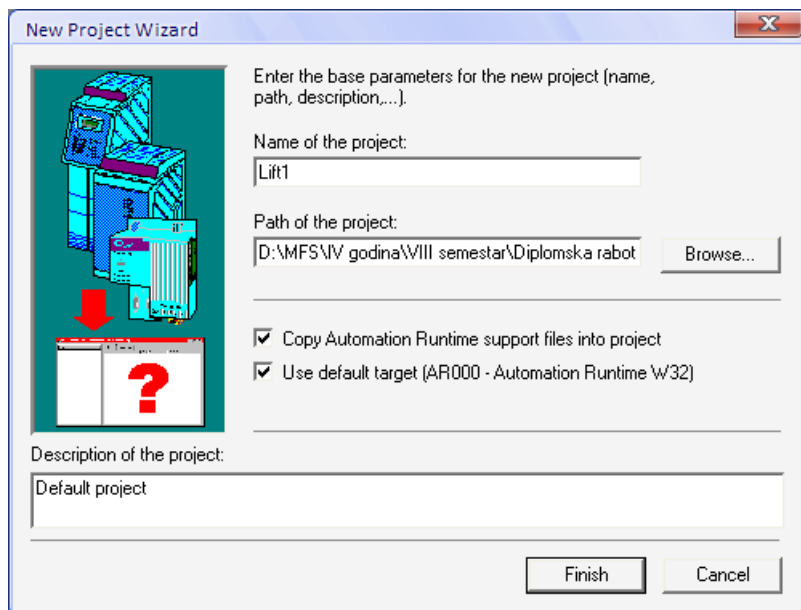
### 6.2.1. Постапка на креирање на проектот

Постапката на креирање нов проект започнува со мениото **File / New Project**. Притоа се отвора прозорецот New Project Wizard како на слика 6.2.

Потребно е да се направат следните подесувања:

- Да се зададе име на проектот (Lift1);
- Да се внесе локацијата на која проектот ќе се зачува (пр. C:\Projects);
- Опцијата **Copy Automation Studio Runtime support files into project** треба да биде штиклирана, што значи дека оперативните системски фајлови ќе бидат зачувани во проектот;

- Треба да се штиклира опцијата **Use default target (AR000 – Automation Runtime W32)**, за да може да се користи симулација на проектот;
- Да се внесе краток опис на проектот (не мора).



Слика 6.2 – Креирање на нов проект

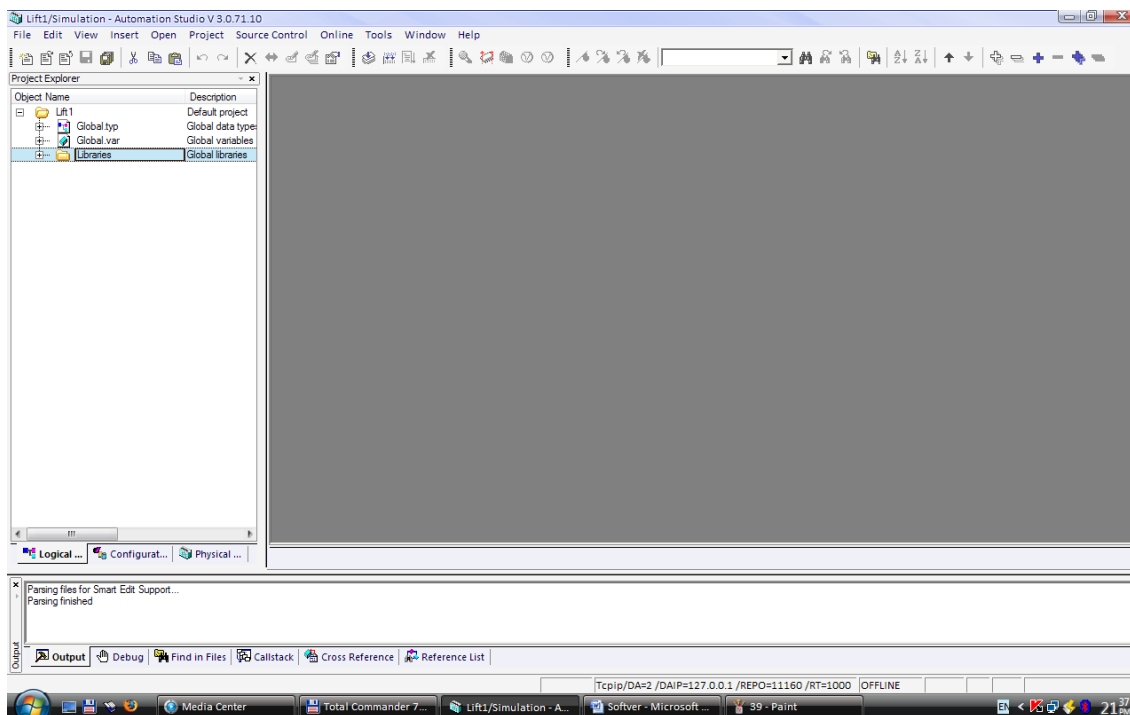
На крај се кликува на копчето **Finish** и проектот е креиран и прозорецот на програмата изгледа како на слика 6.3.

### 6.2.2. Креирање на програма

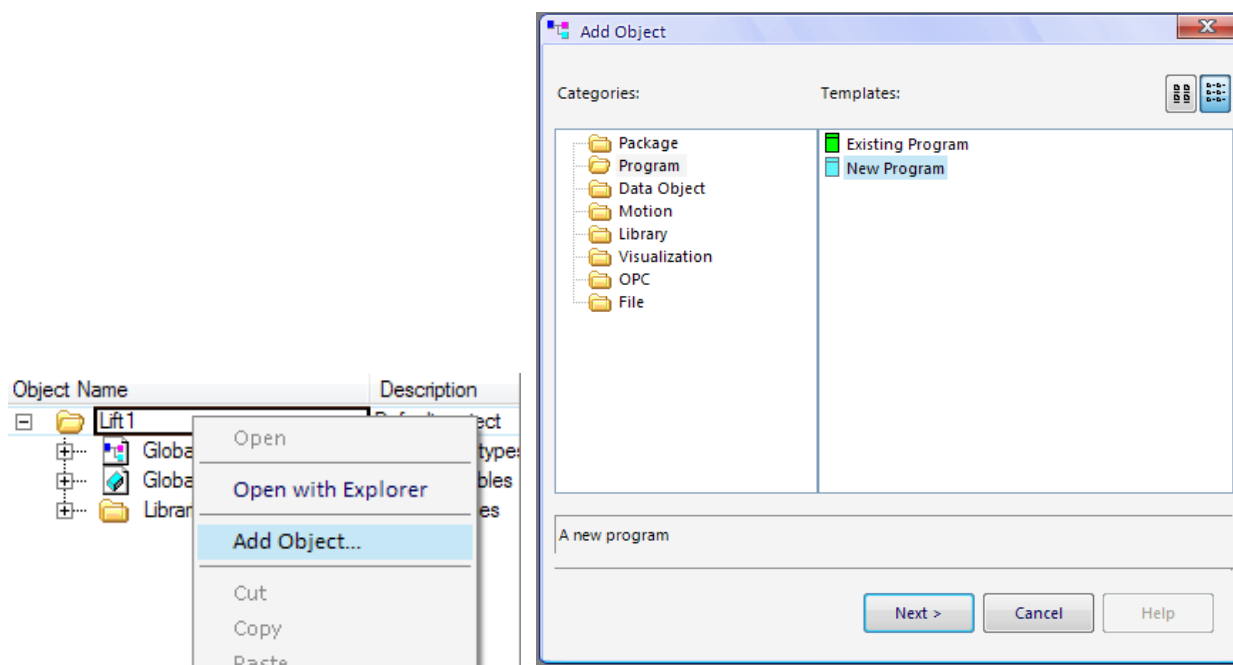
За да се внесе лидер дијаграм во проектот, потребно е да се превземат следните чекори:

- Внесување на лидер дијаграм програмата
- Декларирање на променливите
- Програмирање на лидер дијаграмот

Внесување на лидер дијаграм програма се прави со десен клик на името на проектот (Lift1) во просторот Project Explorer, и се одбира опцијата **Add Object** (слика 6.4). Потоа се појавува дијалог прозорец како на истата слика. Во просторот **Categories** се одбира **Program**, а на десната страна од просторот **Templates** се одбира **New Program** и се кликува на копчето **Next**.



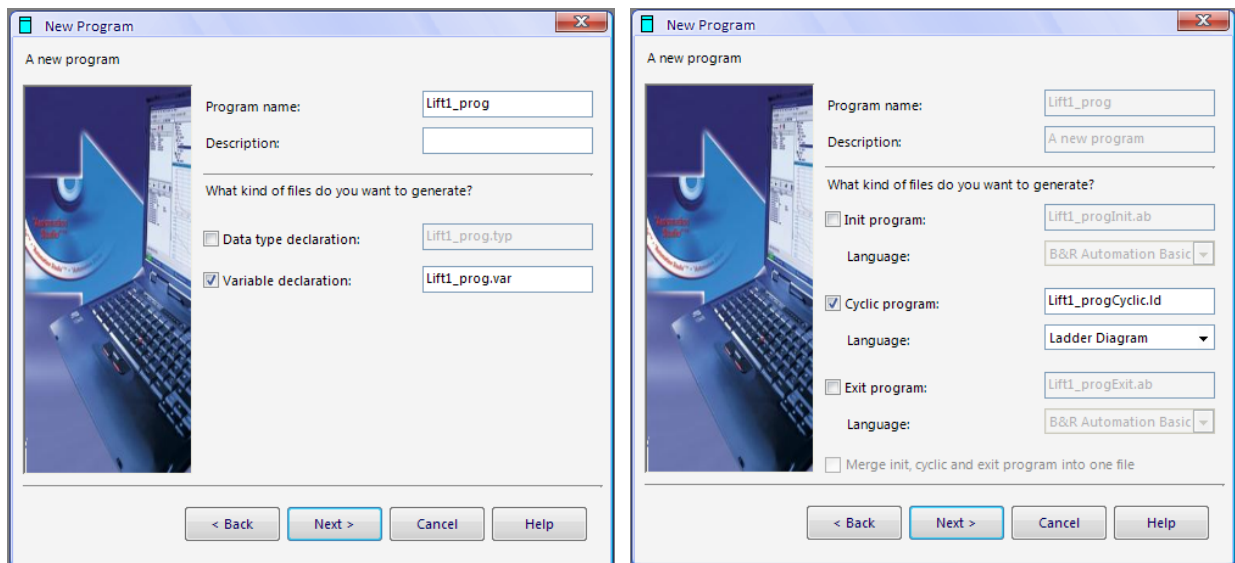
Слика 6.3 – Изглед на нов проект



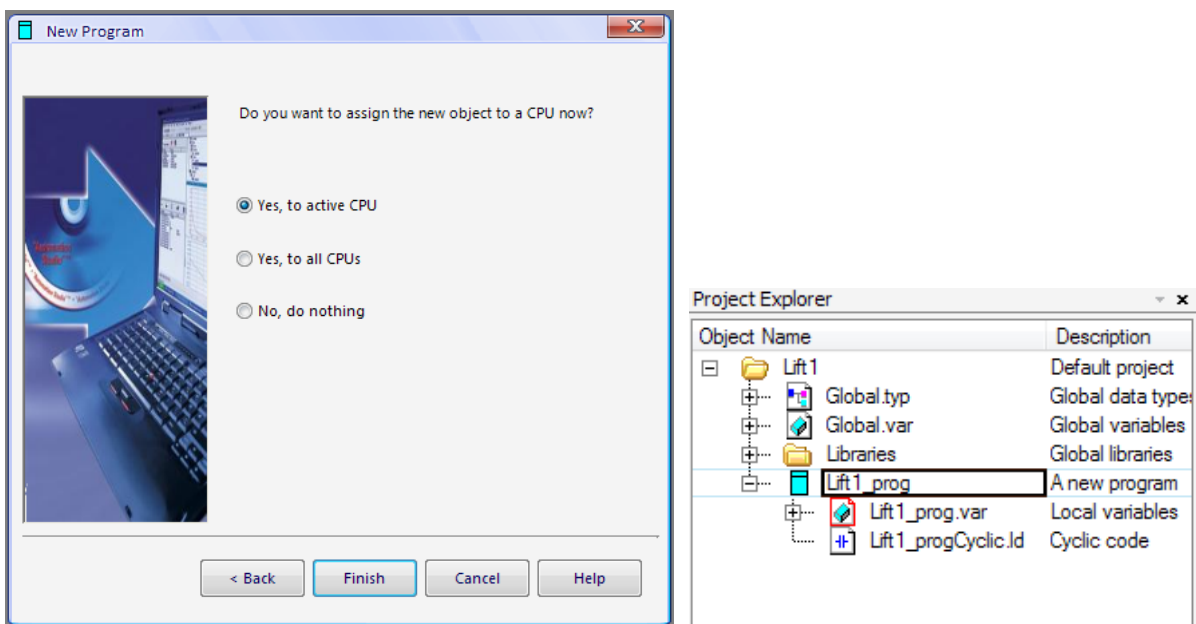
Слика 6.4 – Внесување на лидер дијаграм

Откако ќе се кликне на копчето **Next**, се појавува дијалог прозорец како на слика 6.5 – лево. Овде треба да се внесе име на програмата (**Lift1\_prog**), а исто така може да се внесе и краток опис. Ако е штиклирано полето **Data type declaration** во програмата ќе се генерира фајл, во кој корисникот може да декларира свои типови на податоци што ќе ги содржи програмата. Со штиклирањето на полето **Variable declaration** се генерира фајл за

декларирање на локалните променливи на новата програма. Постапката продолжува со следниот дијалог прозорец (слика 6.5 – десно) со кликување на копчето **Next**. Во следниот дијалог прозорец треба да се селектира програмскиот јазик (Ladder Diagram). Притоа, возможно е да се генерираат три типа на делови од програмата: дел од програмата за иницијализација (Init program), цикличен дел од програмата (Cyclic program) и дел од програмата за излез (Exit program). Сите делови од програмата мора да бидат напишани во ист програмски јазик. Изборот на програмскиот јазик на програмата се случува овде.



Слика 6.5

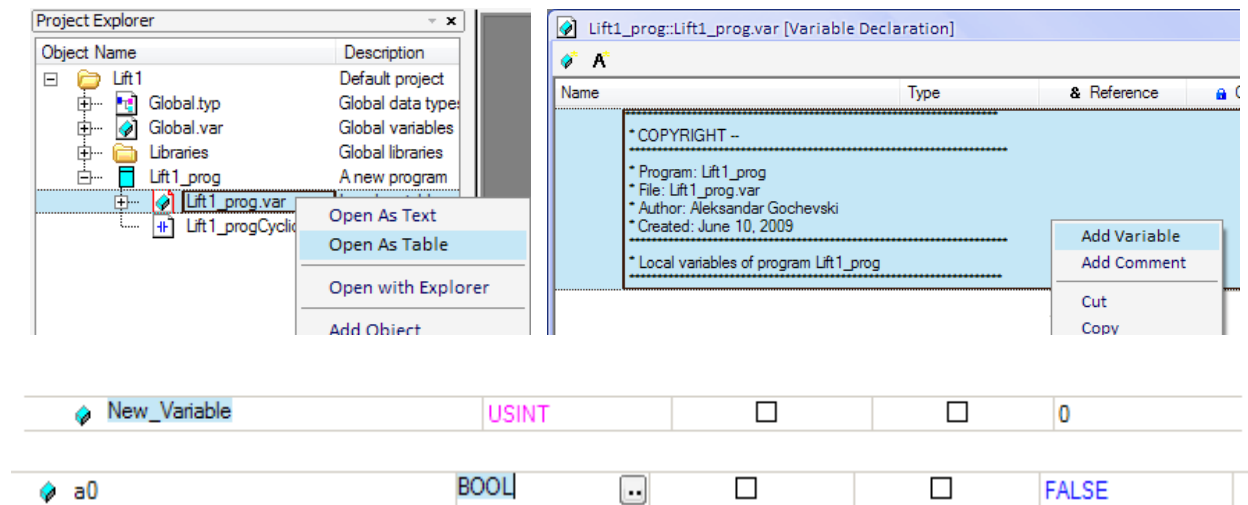


Слика 6.6

После притискање на копчето **Next** се појавува дијалог прозорец како на слика 6.6. Со доделување на објектот на централната процесорска единица, штиклирајќи ја опцијата

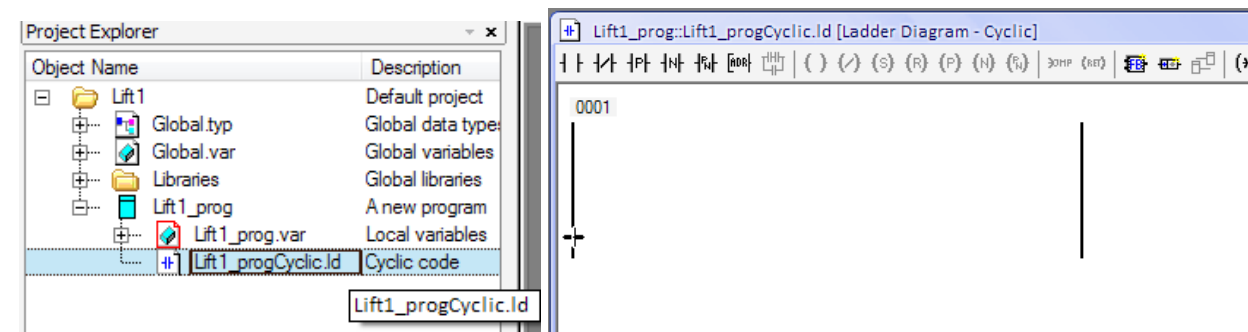
**Yes, to active CPU**, значи дека креираната програма е автоматски доделена на конфигурацијата на софтверот. Со копчето **Finish** завршува постапката на креирање нова програма и во Project Explorer проектот добива изглед како на слика 6.6 (десно).

Во следниот чекор ќе биде опишан начинот како се декларираат променливите во ледер дијаграмот. Променливите се декларираат во фајлот со екстензија `.var` (во случајов `Lift1_prog.var`) со десен клик на фајлот, при што се одбира опцијата **Open As Table** (слика 6.7). Во десната половина од екранот се појавува нов прозорец. На него се кликува со десен клик и се одбира опцијата **Add Variable**, каде што се пишува името на променливата (пр. `a0`) и се одбира типот на променливата (пр. **BOOL**).



Слика 6.7

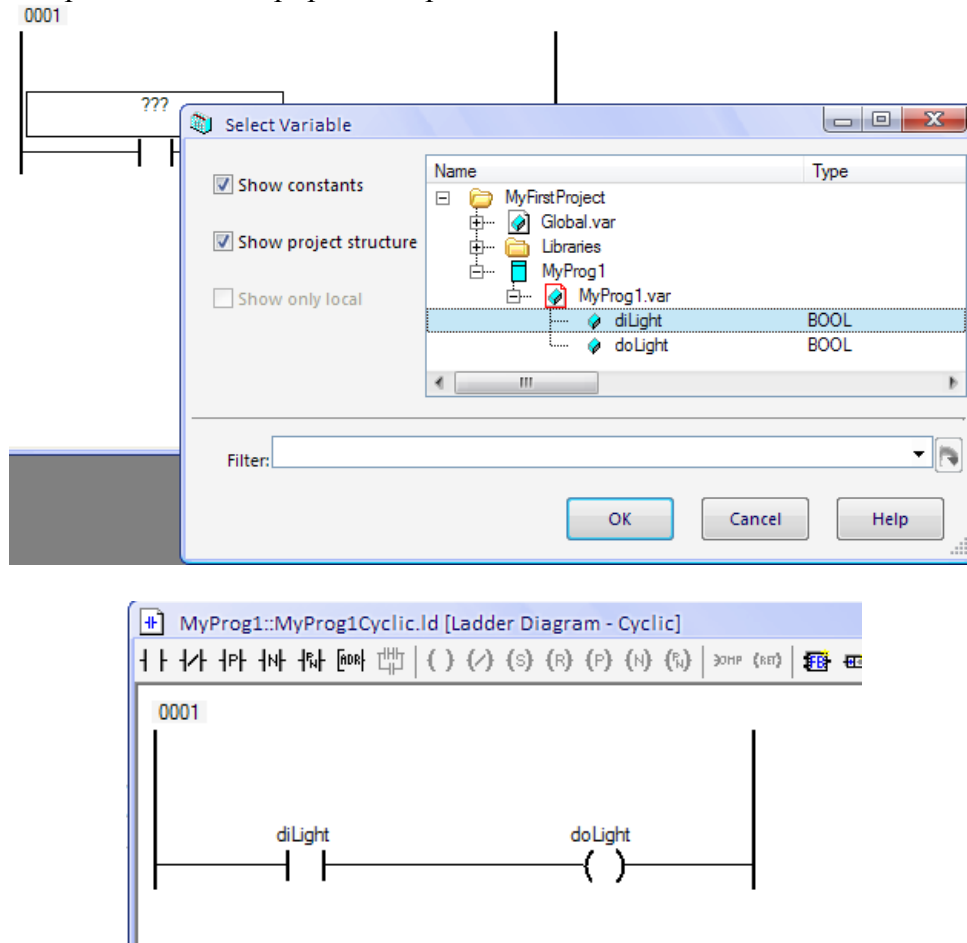
Откако ќе се дефинираат променливите, тие можат да се користат во ледер дијаграмот. За да се програмира ледер дијаграмот, потребно е да се отвори фајлот со екстензија `.ld`, што го содржи зборот `Cyclic` пред точката (`Lift1_progCyclic.ld`) – слика 6.8. Во десната страна на главниот прозорец се појавува едитор за програмирање на ледер дијаграмот.



Слика 6.8

Во едиторот следи програмирањето на ледер дијаграмот. На пример, ако внесеме еден нормално отворен контакт, со притискање на **space bar** – от се отвора прозорец како на слика 6.9, од каде што се одбира која од претходно декларираните променливи ќе се

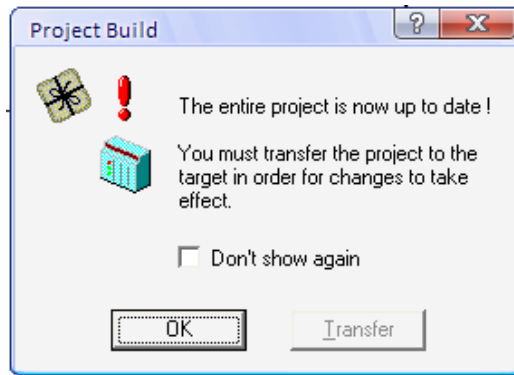
додели на контактот. На слика 6.9 е претставен пример на еден прост ледер дијаграм, користејќи ги претходно декларирани променливи.



Слика 6.9

### 6.2.3. Компајлирање (преведување) на проектот

Откако е креиран проектот и испрограмиран ледер дијаграмот (или програма во друг програмски јазик што е поддржан од Automation Studio), следи компајлирање т.е. преведување на проектот. Со оваа постапка се проверува дали има грешки при програмирањето на програмата. Проектот се компајлира од менито **Project / Build Configuration** или со притискање на копчето **F7** од тастатурата. Откако проектот успешно ќе се компајлира, се појавува прозорец како на слика 6.10. Тој исто така не информира дека проектот може да се пренесе до целниот систем (PLC – то или Automation Runtime за да се изврши симулација).



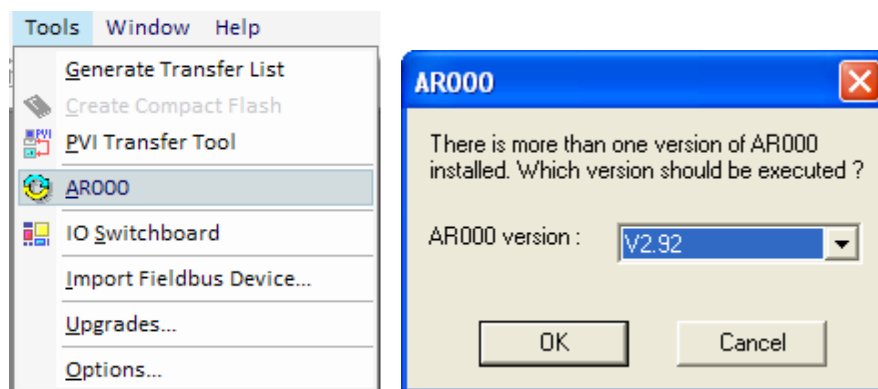
Слика 6.10 – Успешно преведена програма

#### 6.2.4. Пренос на проектот кон целниот систем

Во овој дел ќе се запознаеме како да се дефинира целен систем за еден проект. Во овој дел, наместо вистинското PLC, ќе го користиме софтверот Automation Runtime AR000. Со помош на овој софтвер, може да се изврши симулација на функционирањето на програмата пред таа да се префрли на PLC – то.

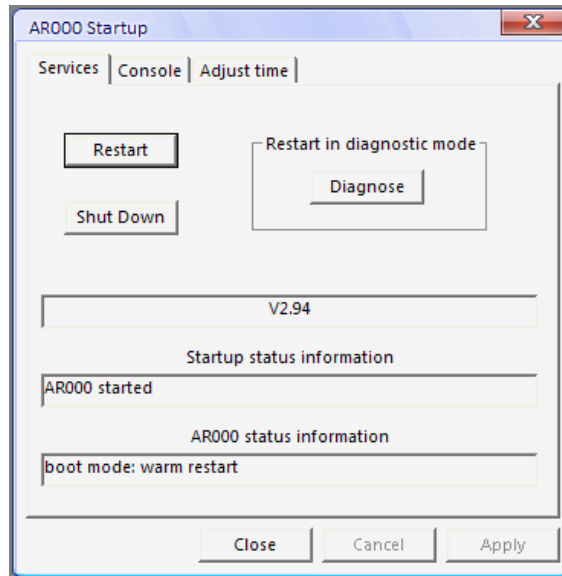
Програмата AR000 претставува извршна програма на автоматскиот систем, која се стартува и работи на обичен персонален компјутер. Таа е базирана на Windows оперативан систем и не е способна за извршување на програмите во реално време, но во основа кореспондира со функционирањето на сите останати целни системи (PLC). Се користи за тестирање на функционирањето на програмите (симулација на програмата без реален управувачки хардвер и без реални извршни компоненти), така што не се потребни физички влезови и излези. Обично AR000 се користи на ист компјутер на кој веќе има инсталирано Automation Studio (обично инсталацијата на Automation Studio содржи и инсталација на AR000), но возможна е комуникација помеѓу AR000 и Automation Studio кога тие се наоѓаат на различни компјутери, со помош на TCP/IP врска.

За да се направи симулација на програма, потребно е првин да се стартува AR000 и проектот да се пренесе во AR000 како целен систем. Стартувањето и преносот на проектот на AR000 всушност заменува (симулира) поврзување на компјутерот и трансфер на проектот во реалниот управувачки хардвер (PLC – то). AR000 се стартува од менито **Tools / AR000** (слика 6.11 – лево). Доколку се инсталирани повеќе верзии на AR000, се појавува прозорецот на слика 6.11 – десно, во спротивно се појавува прозорецот на слика 6.12.




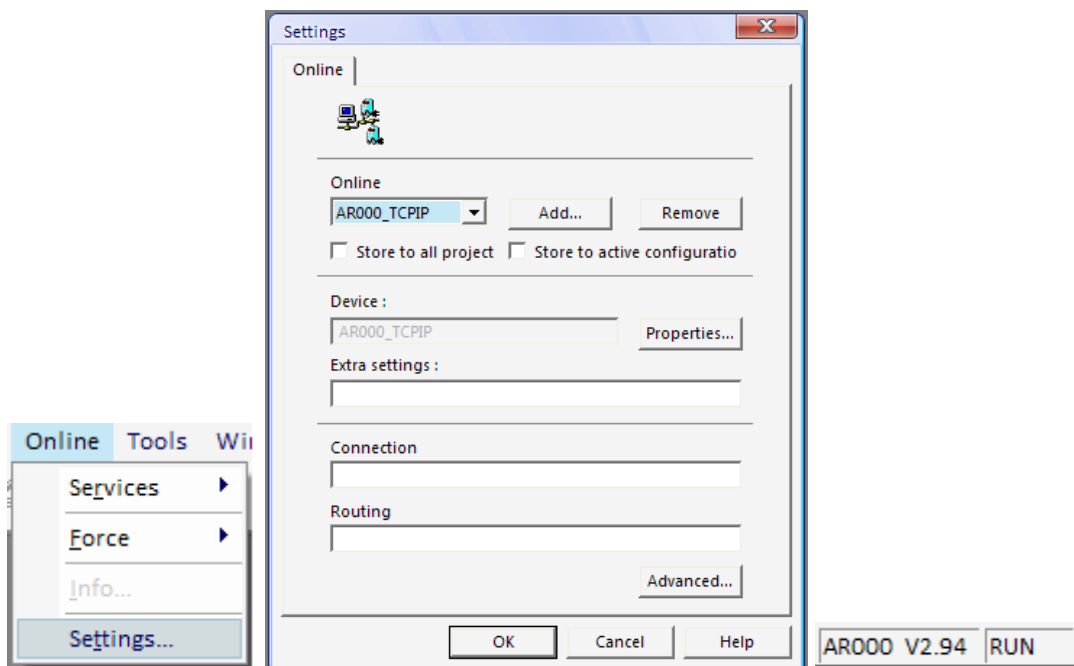
Слика 6.11





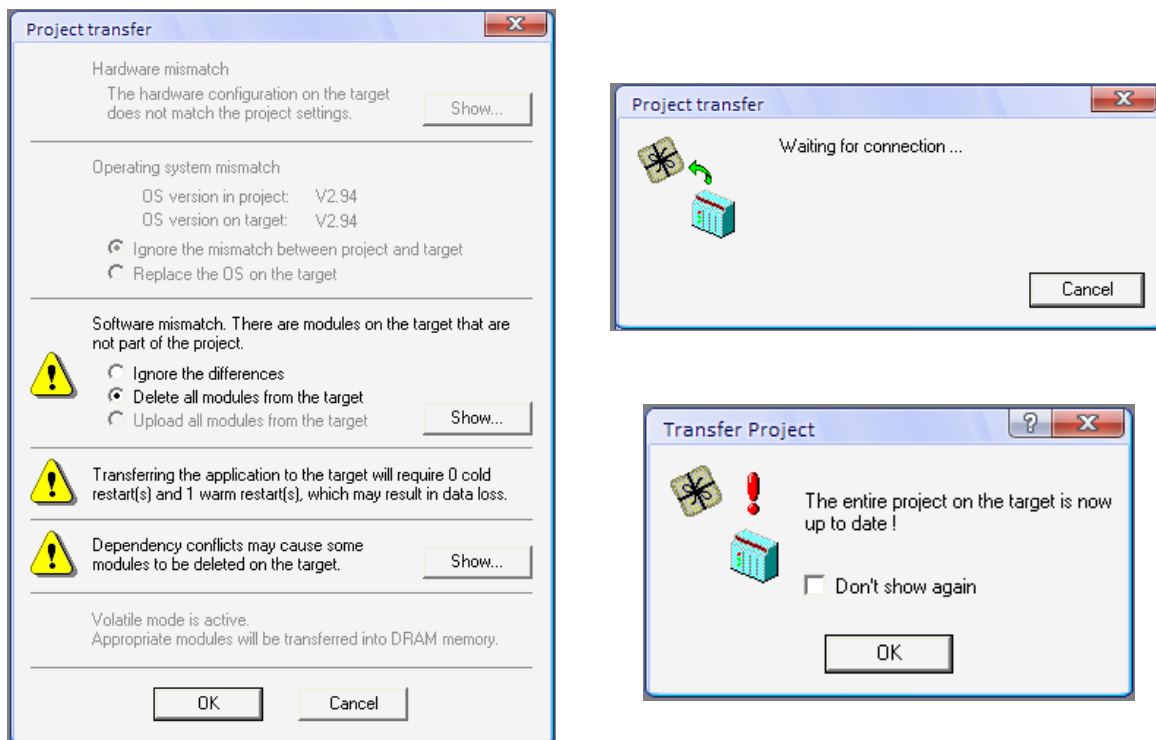
Слика 6.12

Откако ќе се појави прозорецот на слика 6.12, AR000 е стартуван и се појавува неговата икона  во листата со активни програми или процеси (taskbar) во долниот десен дел на мониторот и може да се користи како целен систем. Пред да се префрли проектот во AR000 како целен систем, треба да се воспостави врска со целниот систем. Тое се прави од менито **Online / Settings** (слика 6.13 – лево), по што се појавува прозорец како на слика 6.13 – средина. Подесувањето на конекцијата се прави така што се одбира **AR000\_TCPIP** од полето **Online**. Откако ќе се подигне AR000, на статусната линија (слика 6.13 – десно) наместо **OFFLINE** се појавува **RUN**, што означува дека врската со целниот систем е воспоставена.





Слика 6.13 – Подесување на AR000 за воспоставување на врска

Откако ќе се воспостави врската со целниот систем, следи пренос на проектот во целниот систем. За да се пренесе проектот во целниот систем, потребно е да се зададе командата **Transfer To Target**, од менито **Project / Transfer To Target**, или скратено од тастатура со **Ctrl+F5**. Командата е достапна и на линијата со алатки. Потоа се појавува дијалог прозорец како на слика 6.14.

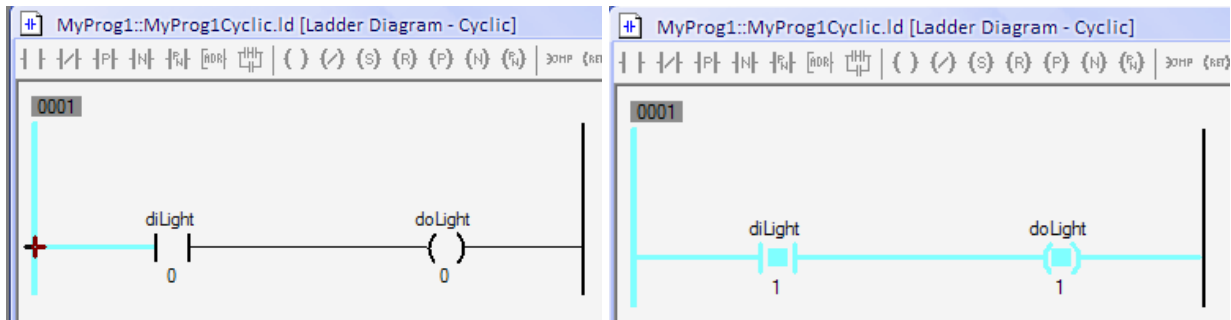


Слика 6.14 – Трансфер на проектот

Овој дијалог прозорец дава информации за постоечки модули на централната процесорска единица (за AR000, таа е виртуелна). Ако постои несогласување помеѓу модулите на софтверот на проектот и целниот систем т.е. постоје на софтверски модули на целниот систем од некој друг проект, претходно префрлен, тој може да се отстрани со одбирање на опцијата **Delete all modules from the target**. Со притискање на копчето OK, започнува трансферот кон целниот систем, а дијалог прозорецот од слика 6.14 – десно не информира дека проектот се наоѓа на целниот систем и е подготвен за извршување. Сега програмата може да се тестира со симулација во AR000 за да се провери дали таа правилно функционира.

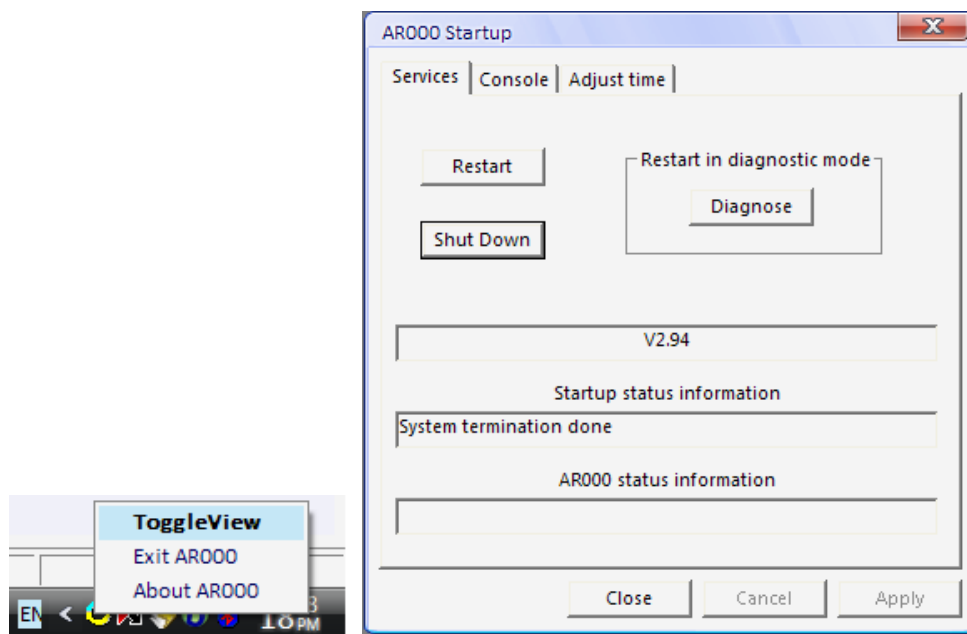
Тестирањето на лидер дијаграмот започнува со активирање на командата Monitor  од линијата со алатки, или со Ctrl+M од тастатура. Сега може да се провери дали со промена на влезните параметри, се добива посакуваниот излез. Текот на сигналите на лидер дијаграмот се вклучува со командата Powerflow  од линијата со алатки, или од менито **Ladder / Powerflow**. Текот на сигналите започнува од левата страна на дијаграмот и се движи кон десната страна (подетално објаснување за текот на сигналите е дадено во поглавјето Ледер програмирање). На пример, нормално отворен контакт не го спроведува сигналот, доколку променливата означена на тој контакт е тип BOOL и има вредност 0

(FALSE). Текот на сигналите се следи со сино обојување на линиите каде што тој поминува (слика 6.15). Корисникот управува со симулацијата, така што ги променува состојбите на влезовите како што би очекувал тие да се променуваат во реалниот процес. Со промена на состојбите на влезовите тие можат да го пропуштаат или да не го пропуштаат сигналот. Кога сигналот ќе дојде до некоја излезна или внатрешна променлива, што се наоѓа последна во еден ред од програмата, таа променлива ќе го пропушти сигналот.



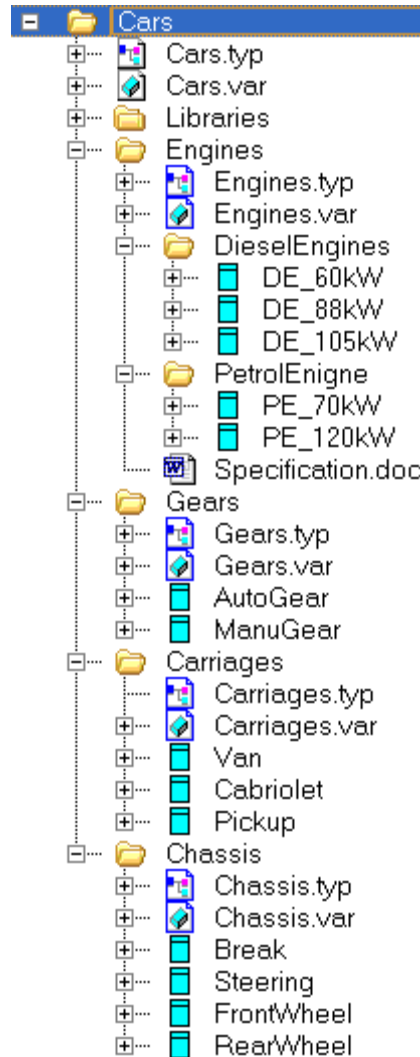
Слика 6.15 – Симулација на програмата со следење на текот на сигналите

Откако ќе се изврши симулација на програмата, потребно е извршната програма на автоматскиот систем AR000 да се исклучи. Излезот од програмата мора да се направи правилно за да се запази конзистентност на податоците. Од линијата со активни процеси (taskbar), се кликува со десен клик на иконата на AR000 и се одбира ToggleView (слика 6.16). Потоа се појавува дијалог прозорец (слика 6.16) и се кликува на копчето **Shut Down**. Овој начин на излез од програмата е најсигурен. Со исклучување на компјутерот или со запирање на процесот на AR000 од Windows Task Manager не се гарантира излез од програмата. Од овој прозорец (AR000 Startup прозорец) може да се изврши и ресетирање на програмата. Нејзиното функционирање е временски ограничено на 2 часа, така што по истекот на ова време потребно е таа да се ресетира.



### 6.3. Основен концепт на Automation Studio 3.0

Создавањето на софтвер со помош на Automation Studio е структурирано така што проектот да наликува на структура на машина. Со ова е овозможено организација и јасен преглед на софтверот. За програмираните машински делови можат да се назначат различни софтверски и хардверски конфигурации. На следната слика 6.17 е прикажан пример на организација на еден проект со помош на овој концепт.



Слика 6.17 – Пример за концептот на Automation Studio

Да претпоставиме дека треба да се произведат три различни типови на возила: комби, пикап и кабриолет. Сите три возила имаат одредени заеднички особини, но истовремено и се разликуваат, на пример, имаат различни мотори, менувачи, шасии и школки. Различните машински делови во проектот (деловите за возилата) треба да ги претставуваат овие компоненти.

На сликата се претставени четири вида на „машински делови“: мотор, менувач, школка и шасија. Тие се лоцирани подлабоко во „дрвото“ на софтверот на Automation Studio. Овие софтверски пакети треба да се искористат за да се поврзат овие различни модели на возила. Највисоко во хиерархијата на „дрвото“ на софтверот е Cars. Под Cars во хиерархијата стојат трите дефиниции: Cars.typ, Cars.var и Libraries. Тие припаѓаат на сите „делови“ на колата (Car).

Во под-дрвото Engines (мотори) стојат дефинициите Engines.typ и Engines.var, кои важат за двата типа на мотори т.е. двата дополнителни подкатегории DieselEngines и PetrolEngines (дизел и бензиски мотори). И двата типа на мотори имаат повеќе подкатегории за различни моќности на моторите.

Во под-дрвото Gears (менувачи) се наоѓаат дефинициите Gears.typ и Gears.var. Тие важат за двете подкатегории на двата типа менувачи, автоматски и мануелен (AutoGear и ManuGear).

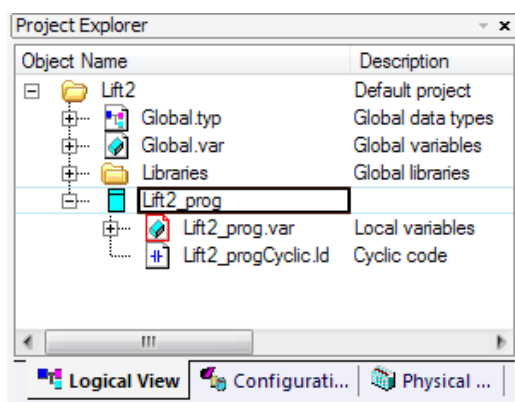
Под-дрвото Carriages (школки) исто така содржи дефиниции Carriages.typ и Carriages.var што се исти за сите типови на школки на возила, но истовремено со посебна функција (посебен софтвер) за секој тип на школка.

Истото важи и за под-дрвото Chassis (шасија), со софтверските подкатегории Brake, Steering, FrontWheel и RearWheel (кочница, управување, преден погон и заден погон).

Овој поглед (дрво) се вика логички поглед (Logical View) на проектот.

### 6.3.1. Логички поглед (Logical View)

Сите софтверски елементи на еден проект се организирани во логичкиот поглед на проектот, во форма на дрво. Елементите на дрвото се папки (фолдери) и објекти. Папките можат да се наречат и пакети. Секој пакет во логичкиот поглед ги претставува софтверот и документацијата за одредениот машински дел. Пакетите можат да се експортираат и импортираат посебно, така што секој член на тимот може самостојно да работи на еден пакет или машински дел. Овде не е даден хардверот на контролерот кој се користи, туку само структурата и распоредот на програмските делови.

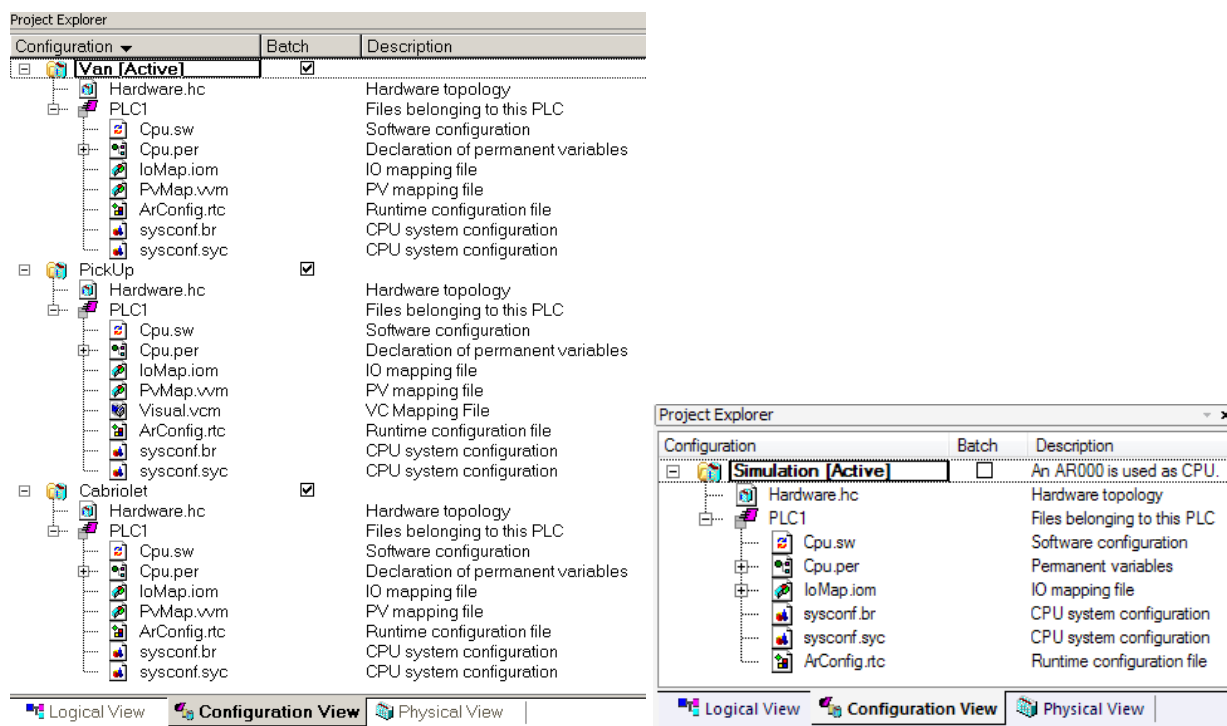


Слика 6.18 – Логички поглед на проектот Lift2

### 6.3.2. Конфигурациски поглед (Configuration View)

За да се овозможи доделување на деловите на одреденото возило (според претходниот пример), Automation Studio го содржи конфигурацискиот поглед. Во него

можат да се креираат, уредуваат, променуваат, бришат и активираат различни конфигурации. Секоја од конфигурациите содржи хардвер и софтвер. Само една конфигурација може да биде активна во дадено време. Активната конфигурација е означена со дебели букви (**bold**) и ја содржи додавката [**Active**]. Во зависност од тоа која конфигурација е активна, оној хардвер што е одреден за конкретната конфигурација е прикажан во физичкиот поглед (Physical View). Сите подесувања за целниот систем можат да се направат во конфигурацискиот поглед. На следната слика 6.19 е даден пример за кондигурациски поглед (за претходниот пример), како и конфигурацискиот поглед за нашиот проект, т.е. за проектот Lift2 за управувањето на хидрауличниот лифт, во почетна форма со само една конфигурација (онаа наменета за симулација).



Слика 6.19 – Конфигурациски погледи

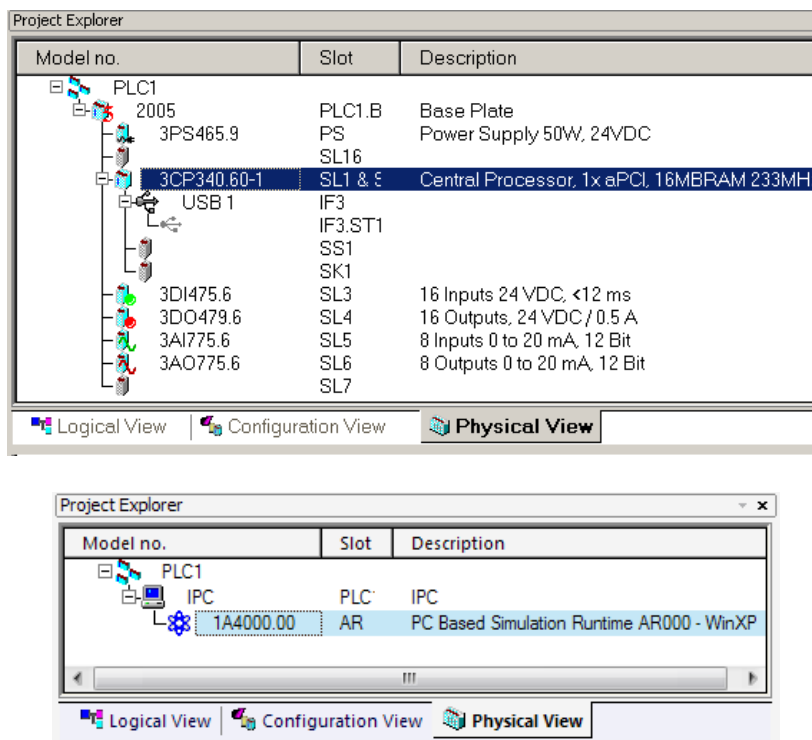
### 6.3.3. Физички поглед (Physical View)

Она дрво со хардвер селектирано (активно) во конфигурацискиот поглед е покажано во физичкиот поглед. Хардверот за активната конфигурација се дефинира и подесува во овој поглед. Во физичкиот поглед можат да се подесат следните работи:

- интерфејс картичките (за online врска, на пример)
- влезно/излезните модули
- одредување на влезно/излезните приклучни точки
- отворање на софтверската конфигурација

На слика 6.20 се прикажани физички погледи за примерот и за проектот за управување на хидрауличниот лифт (Lift2). Во физичкиот поглед на проектот Lift2 нема

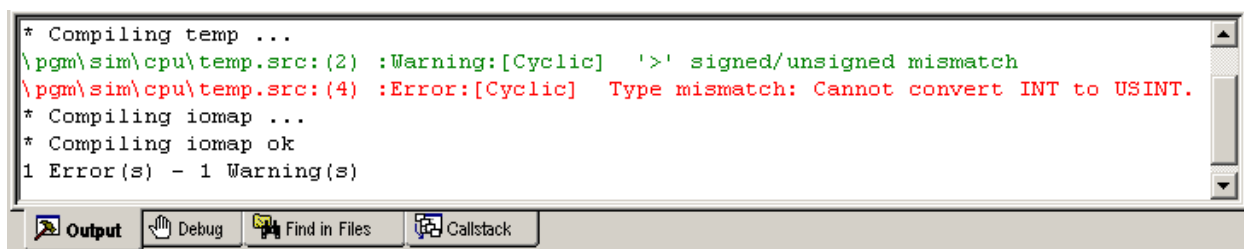
назначено хардвер, а единствениот физички изглед на активната конфигурација Simulation (слика 6.19) е Automation Runtime.



Слика 6.20 – Физички погледи

### 6.3.4. Излезен прозорец

На излезниот прозорец се прикажуваат различни видови на информации. Така, предупредувањата се прикажуваат со зелен текст, грешките со црвен, а информациите со нормален текст. Овие работи се доста важни, особено при откривање на грешки за време на преведувањето (компајлирањето) на програмата.



Слика 6.21 – Излезен прозорец

Во излезниот прозорец се прикажуваат:

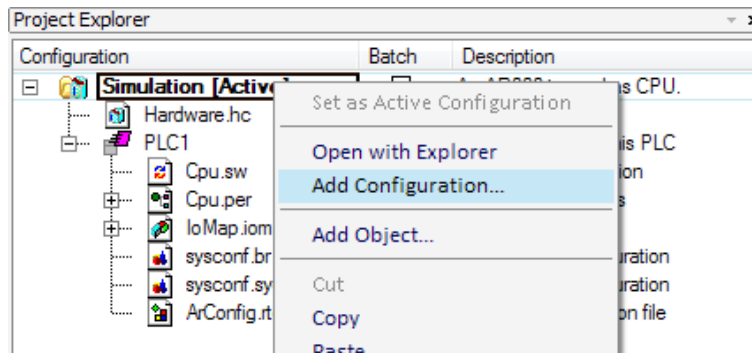
- Пораки на предупредувања и грешки за време на преведувањето на програмата. Двоен клик на пораката го носи курсерот на оној ред од програмата каде што се наоѓа грешката.
- Прогресот и статусот при пренесување на проектот до целниот систем.

- Пораки за време на внесување или бришење на објекти во проектот или на целниот систем.
- Излезен прозорец за пораките на дебагерот.
- Излез за резултатите на функцијата “Find in Files”, која пребарува по сите фајлови во проектот.

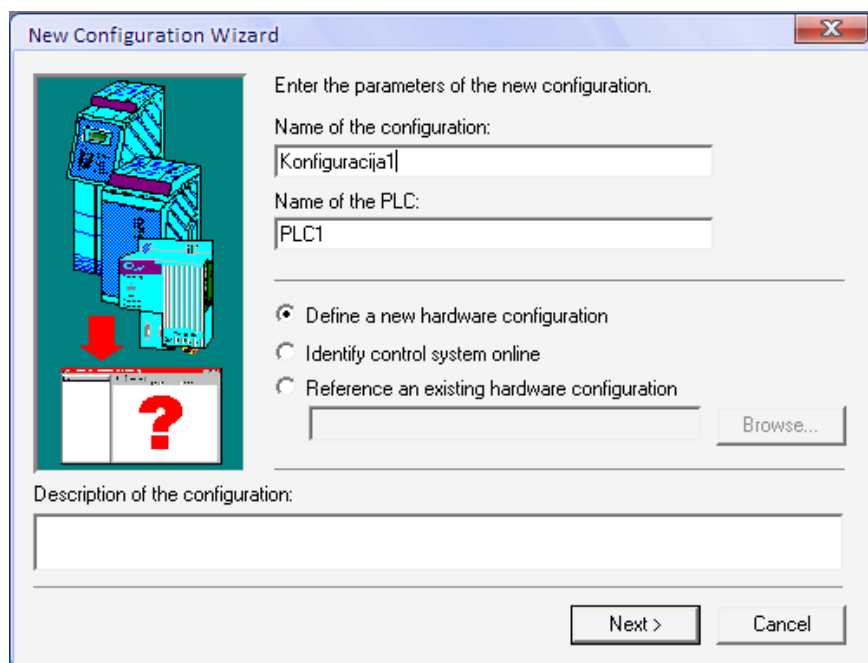
## 6.4. Подесувања на софтверот и хардверот во еден проект

### 6.4.1. Креирање на нова конфигурација

Креирањето на нова конфигурација започнува со отворање на конфигурацискиот поглед (Configuration View). На нашиот проект Lift2, моменталниот конфигурациски поглед изгледа како на слика 6.19 – десно, т.е. содржи само конфигурација за симулација. Со десен клик во полето на Project Explorer (слика 6.22) се одбира опцијата **Add Configuration** и се појавува прозорец како на слика 6.23.



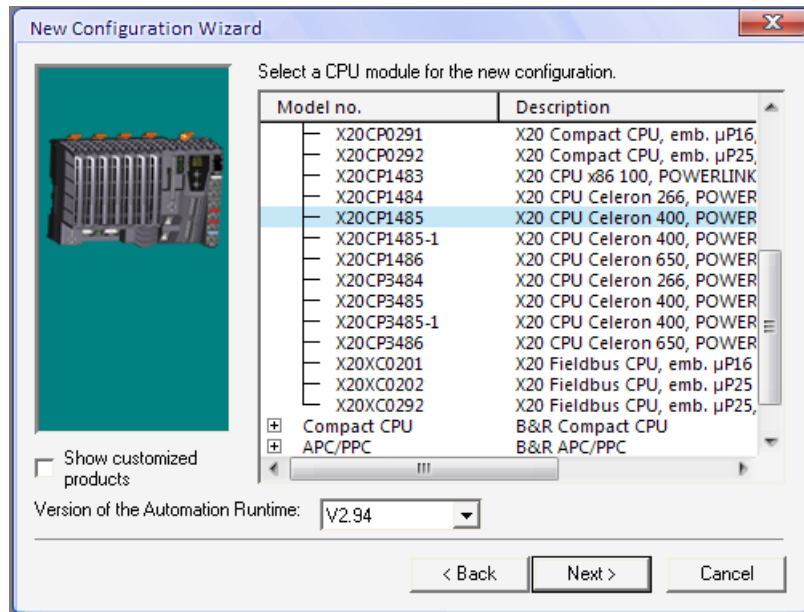
Слика 6.22



Слика 6.23

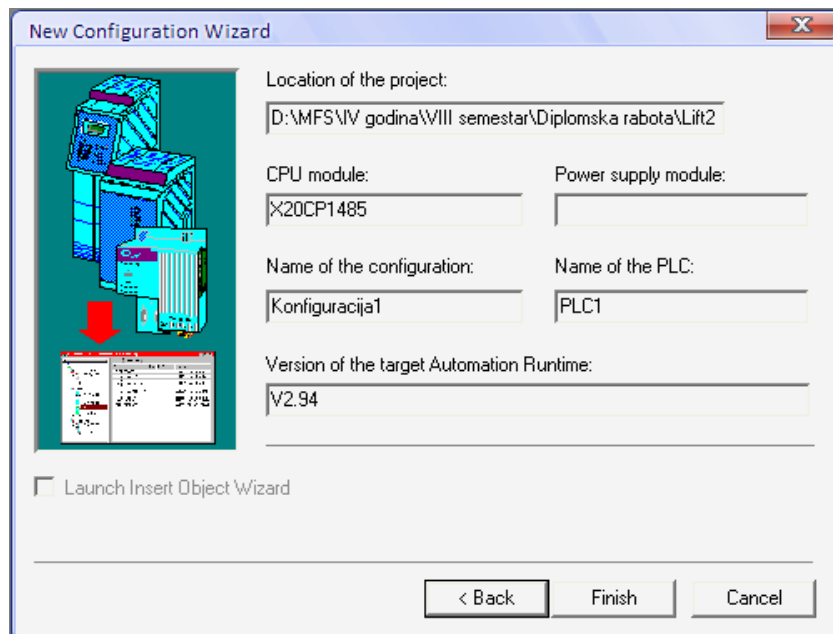


Во прозорецот како на слика 6.23 се внесува името на новата конфигурација (Konfiguracija1) и се одбира опцијата **Define a new hardware configuration**, што значи дека дефинираме нова хардверска конфигурација. Потоа се продолжува со постапката на креирање на нова конфигурација со кликување на копчето **Next**. Следен чекор е изборот на централната процесорска единица (CPU). За нашиот проект тоа ќе биде процесорот со кој располага PLC-то на нашата лабораторија - X20 CP1485. Потоа се кликува **Next**.



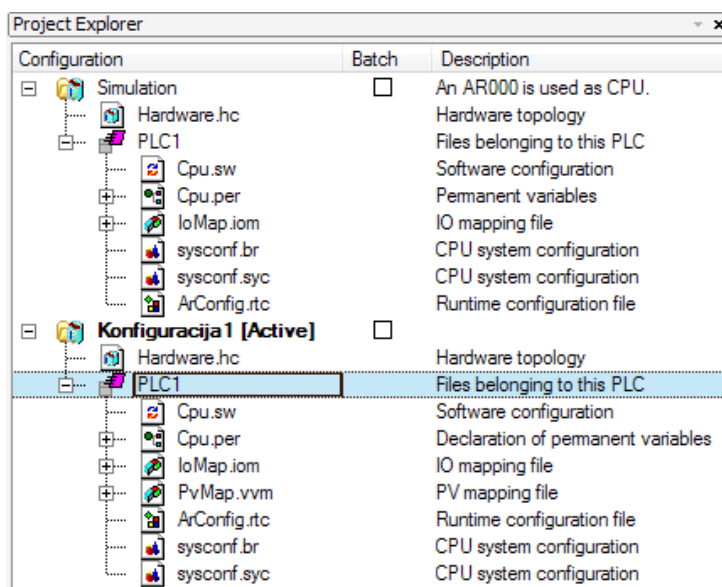
Слика 6.24

На следниот прозорец (слика 6.25) се прикажани сумирани опциите кои претходно сме ги избрале. Креирањето на конфигурацијата го завршуваме со кликување на копчето **Finish**.



Слика 6.25

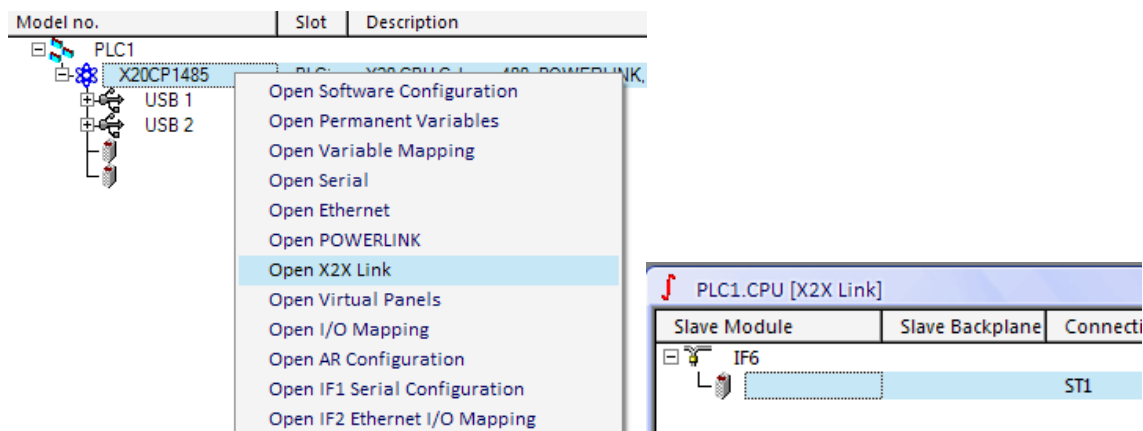
Новокреираната конфигурација „Konfiguracija1“ е успешно креирана. Конфигурацискиот поглед сега изгледа како на слика 6.26. Со двоен клик на името на конфигурацијата може да се менува активната конфигурација.



Слика 6.26

#### 6.4.2. Додавање на потребниот хардвер

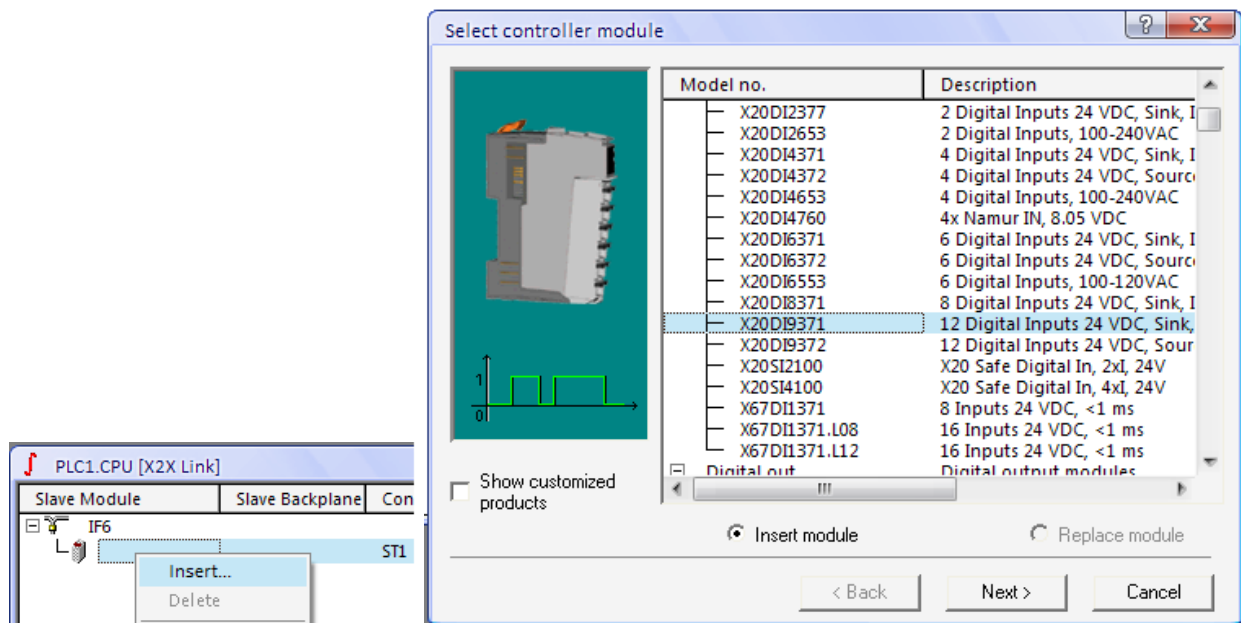
Хардверот што го содржи активната конфигурација е прикажан во физичкиот поглед (Physical View). Тука се додава хардверот потребен за реализација на задачата од автоматизација, во конкретниот случај, автоматизација на хидрауличниот лифт. Потребно е да се додадат следните модули: температурен модул X20 AT2222, аналоген влезен модул X20 AI4622, аналоген излезен модул X20 AO4622, дигитален влезен модул X20 DI9371 и дигитален излезен модул X20 DO9322. Тоа се модулите што се приклучени на контролерот по наведениот редослед. Со десен клик на ознаката на CPU-то (X20 CP1485) се отвора мени како на слика 6.27 – лево.



Слика 6.27 – Отворање на X2X врската на X20 CP1485

Тука се избира видот на врската помеѓу централната процесорска единица и модулите, а тоа е X2X врската. Откако ќе се кликне на **Open X2X Link**, се отвора прозорец како на слика 6.27 – десно и следи подесувањето на интерфејсниот приклучок IF6 (слика 2.6).

На IF6 приклучокот се поврзуваат сите модули што се додаваат кон CPU-то. Затоа во тој прозорец се кликува со десен клик и се избира **Insert** (слика 6.28). Од прозорецот на слика 6.28 – десно се избираат сите модули што треба да се додадат, повторувајќи ја постапката за секој модул одделно.



Слика 6.28 – Селекција на потребните модули

Откако заврши додавањето на хардверот, физичкиот поглед на проектот Lift2 изгледа како на слика 6.29.

Model no.	Slot	Description
PLC1		
X20CP1485	PLC1.CPU	X20 CPU Celeron 400, POWERLINK, 1x IF
X20AT2222	PLC1.CPU.IF6.ST1	2 Resistor Temperature Inputs
X20AI4622	PLC1.CPU.IF6.ST2	4 Inputs $\pm 10$ V / 0 to 20 mA
X20AO4622	PLC1.CPU.IF6.ST3	4 Outputs $\pm 10$ V / 0 to 20 mA
X20D19371	PLC1.CPU.IF6.ST4	12 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
X20DO9322	PLC1.CPU.IF6.ST5	12 Outputs 24 VDC / 0.5 A

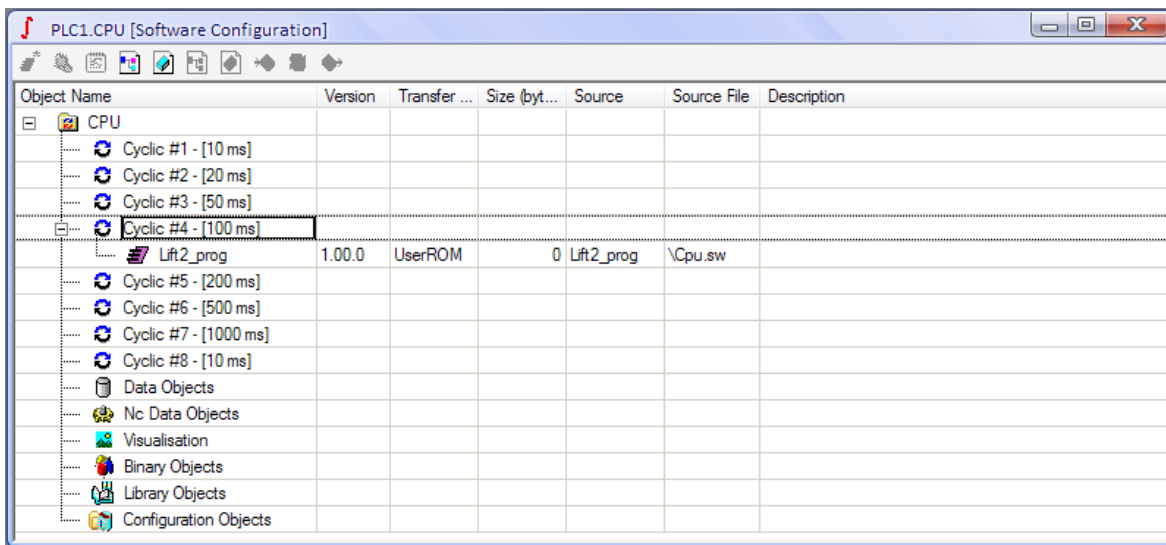
Слика 6.29

### 6.4.3. Софтверска конфигурација и доделување на софтверот

Откако е конфигуриран потребниот хардвер, потребно е додели софтверот на избраниот хардвер. Софтверската конфигурација за проектот се отвора со двоен клик на името на CPU-то во физичкиот поглед и се отвора во десната страна на главниот прозорец. На ова место се подесуваат софтверските елементи (програмите) на соодветните циклуси,

т.е. се избира времетраењето на еден циклус во програмата. Времетраење на циклус е она време кое е потребно да се скенираат состојбите на сите влезови и да се дадат соодветните сигнали на излезите, според поставената логика. Тоа време, како и некои други важни параметри поврзани со циклусите можат да се подесат.

За да се доделат програмите, потребно е да се отвори логичкиот поглед, додека на десната страна од главниот прозорец е отворена софтверската конфигурација. Се селектира името на програмата (Lift2\_prog) и се влече во посакуваниот циклус. Во конкретниот случај програмата за лифтот ќе ја поставиме во циклусот број 4, со времетраење од 100 ms (слика 6.30).



Слика 6.30 – Софтверска конфигурација

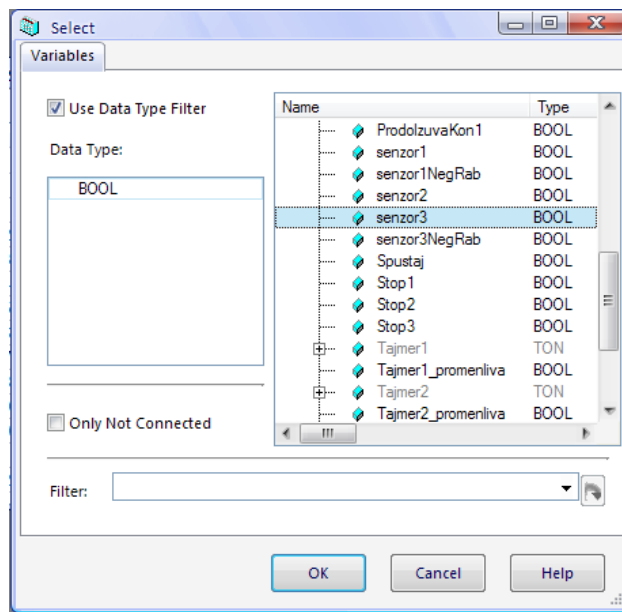
На истиот начин може да се додели било кој софтверски објект од логичкиот поглед во софтверската конфигурација. Овде е потребно да се забележи дека само програми од активната конфигурација можат да се доделат на софтверската конфигурација. Програмите стануваат управувачки откако ќе се доделат на софтверската конфигурација.

#### 6.4.4. Доделување на променливите на влезно/излезните модули

Откако ќе се подесат хардверските и софтверските параметри, потребно е сите променливи во програмата да се доделат на соодветните влезно/излезни модули. Со овој чекор, всушност, се означува која променлива е влезна, а која излезна. При доделувањето на променливите на влезно/излезните модули, секоја променлива се доделува на точно определен со број приклучок на модулот.

Во нашиот проект има два модули, дигитален влезен X20 DI9371 и дигитален излезен X20 DO9322. Доделувањето за секој модул е одделно и започнува со отворање на прозорецот **I/O Mapping**, со двоен клик на соодветниот модул (слика 6.31). Во овој прозорец, за секој со број означен влез/излез се доделуваат променливите. Подоцна, при физичкото поврзување на влезовите и излезите со реалните хардверски уреди, мора да се запази приклучувањето дефинирано овде.

Channel Name	Data Type	Task Class	PV or Channel Name	Inverse	Source File	Description [1]
ModuleOk	BOOL			<input type="checkbox"/>		Module status (1 = module present)
DigitalInput01	BOOL	Automatic	Lift2_prog.a1	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput02	BOOL	Automatic	Lift2_prog.b1	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput03	BOOL	Automatic	Lift2_prog.c1	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput04	BOOL	Automatic	Lift2_prog.k1	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput05	BOOL	Automatic	Lift2_prog.k2.dole	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput06	BOOL	Automatic	Lift2_prog.k2.gore	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput07	BOOL	Automatic	Lift2_prog.k3	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput08	BOOL	Automatic	Lift2_prog.senzor1	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput09	BOOL	Automatic	Lift2_prog.senzor2	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput10	BOOL	Automatic	Lift2_prog.senzor3	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput11	BOOL	Automatic	Lift2_prog.senzorVrata	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink
DigitalInput12	BOOL	Automatic	Lift2_prog.senzorTezina	<input type="checkbox"/>	\NoMap.iom	24 VDC, 0.1 to 25 ms switching delay, sink



Channel Name	Data Type	Task Class	PV or Channel Name	Inverse	Source File	Description [1]
ModuleOk	BOOL			<input type="checkbox"/>		Module status (1 = module present)
DigitalOutput01	BOOL	Automatic	Lift2_prog.Podignuvaj	<input type="checkbox"/>	\NoMap.iom	24 VDC / 0.5 A, source
DigitalOutput02	BOOL	Automatic	Lift2_prog.Spustaj	<input type="checkbox"/>	\NoMap.iom	24 VDC / 0.5 A, source
DigitalOutput03	BOOL	Automatic	Lift2_prog.Vrata1	<input type="checkbox"/>	\NoMap.iom	24 VDC / 0.5 A, source
DigitalOutput04	BOOL	Automatic	Lift2_prog.Vrata2	<input type="checkbox"/>	\NoMap.iom	24 VDC / 0.5 A, source
DigitalOutput05	BOOL	Automatic	Lift2_prog.Vrata3	<input type="checkbox"/>	\NoMap.iom	24 VDC / 0.5 A, source
DigitalOutput06	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
DigitalOutput07	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
DigitalOutput08	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
DigitalOutput09	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
DigitalOutput10	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
DigitalOutput11	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
DigitalOutput12	BOOL			<input type="checkbox"/>		24 VDC / 0.5 A, source
StatusDigitalOutput01	BOOL	Automatic	Lift2_prog.Podignuvaj	<input type="checkbox"/>	\NoMap.iom	Status digital output 01 (0 = OK)
StatusDigitalOutput02	BOOL	Automatic	Lift2_prog.Spustaj	<input type="checkbox"/>	\NoMap.iom	Status digital output 02 (0 = OK)
StatusDigitalOutput03	BOOL	Automatic	Lift2_prog.Vrata1	<input type="checkbox"/>	\NoMap.iom	Status digital output 03 (0 = OK)
StatusDigitalOutput04	BOOL	Automatic	Lift2_prog.Vrata2	<input type="checkbox"/>	\NoMap.iom	Status digital output 04 (0 = OK)
StatusDigitalOutput05	BOOL	Automatic	Lift2_prog.Vrata3	<input type="checkbox"/>	\NoMap.iom	Status digital output 05 (0 = OK)
StatusDigitalOutput06	BOOL			<input type="checkbox"/>		Status digital output 06 (0 = OK)

Слика 6.31 – Доделување на променливите на влезно/излезните модули

При доделувањето на променливите на дигиталниот излезен уред, тие треба да се доделат на две места во прозорецот на слика 6.31 (најдолу): на пример, променливата Podignuvaj треба да се додели на местото DigitalOutput01 и на StatusDigitalOutput01. Доколку променливите се доделат само на местата DigitalOutput, при зачувувањето на фајлот ќе се појави порака во која стои дека доделувањето на променливите не е потполно.

## 7. Програмирање на програмибилниот контролер

Во ова поглавје ќе бидат претставени можностите на V&R контролерите од аспект на нивното програмирање. Ќе бидат презентирани програмските јазици, работата со променливи и константи и ќе бидат дадени основите на Ледер дијаграм програмскиот јазик.

### 7.1. Програмски јазици поддржани кај V&R контролерите и нивни можности

Програмскиот пакет Automation Studio, како околина за програмирање на V&R контролерите, нуди повеќе програмски јазици кои можат да се искористат. При тоа, програмерот може да се одлучи за еден од програмските јазици, но може да користи и повеќе програмски јазици во еден проект, доколку тоа е неопходно. Тие се:

- Ледер дијаграм – Ladder diagram (LD)
- Функциски блок дијаграм – Function block diagram (FBD)
- Дијаграм на непрекината функција – Continuous function Chart (CFC)
- Дијаграм на секвенцијална функција – Sequential function Chart (SFC)
- Листа на инструкции – Instruction list (IL)
- Структуриран текст – Structured text (ST)
- Automation Basic (SB)
- ANSI C (C)

Првите три програмски јазици се графички, дијаграмот на секвенцијална функција е комбиниран – графички и текстуален, а преостанатите четири се текстуални јазици. Во Automation Studio сите текстуални програмски јазици користат ист едитор за пишување на програмите. Алатките за дијагностицирање се исти и се користат на ист начин. Во Watch прозорецот, каде што се проверуваат и поставуваат вредностите на променливите, се користи идентично без разлика на програмскиот јазик. Функциските блокови од стандардните V&R библиотеки можат да се повикуваат и користат кај сите програмски јазици.

Секоја посакувана апликација може да се изведе со користење на било кој од програмските јазици. Во следната табела се дадени можностите на програмските јазици за реализација на различни функционални групи.

	LD	FBD	CFC	SFC	IL	ST	AB	C
Логика	√	√	√	√	√	√	√	√
Аритметика					√	√	√	√
Одлуки	√	√	√	√	√	√	√	√
Повторувачки циклуси						√	√	√
Чекорни секвенци				√		√	√	√
Динамички променливи						(√)	√	√

Функцииски блокови	√	√	√	√	√	√	√	√
--------------------	---	---	---	---	---	---	---	---

**Забелешка:** Со користење на функцииски блокови се овозможува реализација на оние функциии кои не се поддржани кај одреден програмски јазик.

Изработката на програмата која ќе ја управува работата на лифтот ќе биде реализирана во програмскиот јазик ледер дијаграм.

## 7.2. Програмирање во ледер дијаграм програмскиот јазик

Програмскиот јазик ледер дијаграм е графички јазик, кој е најзастапен при програмирањето на програмибилните логички контролери.

### *Краток историјат*

Оригиналниот концепт на програмибилниот логички контролер (PLC) е развиен во САД во 1968 година. Тој концепт е развиен како микропроцесорски и програмибилен, кој требало да ги замени хардверските системи за управување. PLC контролерите биле базирани околу ледер дијаграмот, што претставува шематски приказ на логички управувачки систем од релејни кола. Во тоа време, овој концепт овозможи со релативно малку тренинг, брзо составување и програмирање на прости логички управувачки системи. Ледер дијаграмот е идеален за прости логички управувачки системи, лесно се користи и лесно се совладува. Тој е веројатно главната причина поради која PLC контролерите доживеаја таков успех во индустријата.

До почетокот на 90-тите постоеле илјадници производители на PLC контролери, и сите нуделе свои системи за програмирање и сет на инструкции. Иако програмите што се пишувале за различни системи биле слични, начинот на кој тие биле структурирани, како и сетот на инструкции кој се користел, се разликувал од еден производител до друг. Поради ова секогаш се појавувале проблеми својсвени само за специфичниот хардвер кој се користел, и корисниците биле приморани да се врзуваат за одреден производител.

Во 1979 година била формирана работна група на Меѓународниот комитет за електротехника (IEC – International Electrotechnical Commission) со цел да изготви еден универзален стандард за PLC контролерите. Овој стандард денес е познат како IEC 61131.

### 7.2.1. Општи карактеристики на ледер дијаграмот

Ледер дијаграмот е графички програмски јазик. Тој претставува симболично претставување на електронските кола. Симболите се така избрани, да изгледаат слично со шематските симболи кои се користат кај електричните уреди. Заради ова, еден електротехничар кој никогаш не работел со PLC, може да разбере ледер дијаграм. Шематските симболи (контакти и намотки) и линиите кои ги поврзуваат ја создаваат логиката на програмата. Општи карактеристики се:

- графички програмски јазик
- сличен со релејните дијаграми
- јасно и едноставно програмирање



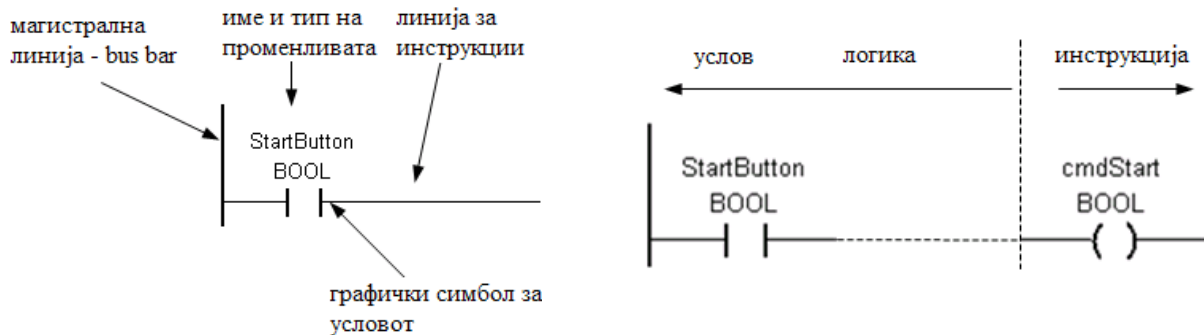
- интуитивен за употреба
- лесно се наоѓаат грешки
- се преведува (компајлира) е со IEC 61131-3 стандардот.

Програмскиот јазик ледер дијаграм, во Automation Studio, ги нуди следните можности:

- Користење на дигитални влезови и излези и внатрешни Булови променливи (тип Boolean)
- Користење на аналогни влезови и излези
- Користење на функциски блокови
- Контрола на текот на програмата (скокови)
- Алатки за дијагностика

### 7.2.2. Основни елементи на ледер дијаграмот

Основните елементи на еден ледер дијаграм се прикажани на слика 7.1. Тие можат да се поделат на два дела: условен дел и дел со инструкции. Во условниот дел се содржани условите кои треба да бидат исполнети за да се изврши инструкцијата која се наоѓа на десната страна од условниот дел, и со која е поврзана со линијата за инструкции. Левата вертикална линија се вика магистрална линија или bus bar. Тоа е линија низ која (замислено) програмата константно се „снабдува“ со напојување. Во условниот дел се наоѓаат графичките симболи за условите т.е. контактите. Логичката комбинација од условите одредува кога и како ќе се изврши инструкцијата на десната страна. Елементите на крајната десна страна се викаат намотки (на пример, светилки, мотори, релиња, итн.)



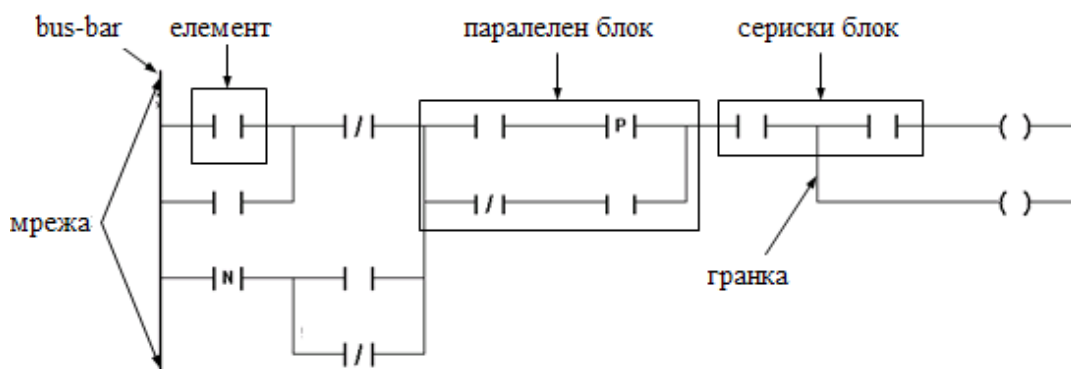
Слика 7.1 – Основни елементи на ледер дијаграмот

### 7.2.3. Мрежи во ледер дијаграмот

Мрежа во ледер дијаграмот е коло со кое се изразува одредена функција. Таа се состои од елементи, гранки и блокови. Со мрежата се изразува некоја целосна функција и таа е основен дел на ледер дијаграмот. Еден целосен ледер дијаграм е составен од повеќе вакви програмски мрежи.

Почетокот на мрежата е на левата вертикална линија bus bar. Ако две или повеќе кола се поврзани со вертикална линија, тогаш тие припаѓаат на иста мрежа. Во една мрежа

можат да се сместат до 50 редови и 50 колони. Големината на ледер дијаграмот е ограничена само од големината на меморијата на компјутерот каде што се програмира и од меморијата на контролерот.



Слика 7.2 – Програмска мрежа на ледер дијаграмот

## 7.2.4. Символи кај ледер дијаграмот

### 7.2.4.1. Контакти

Контактите се наоѓаат во условниот дел од ледер дијаграмот. Тие не можат да бидат поставени на десната страна – таа е резервирана за намотките. Контактите можат да се поврзат со дигиталите влезови или излези на функционалните блокови.

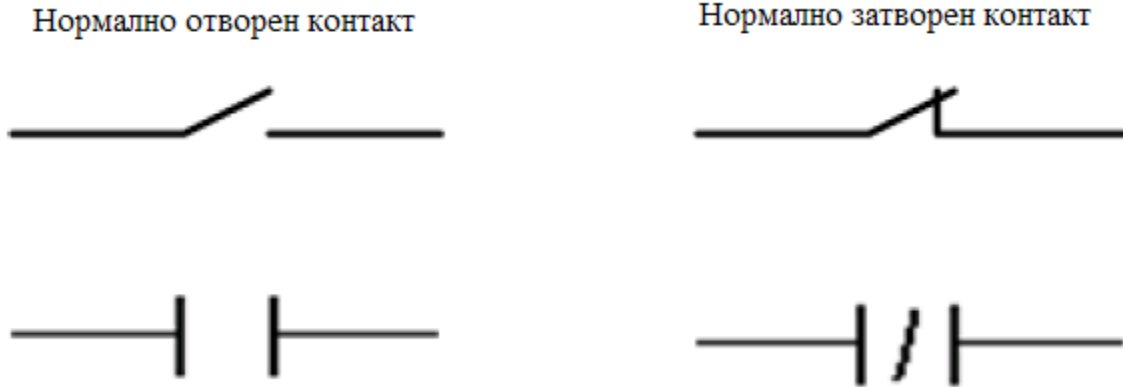
Контактите во една мрежа можат да бидат поврзани со еден или повеќе намотки. Секој контакт е означен со име на променлива, која треба да биде декларирана во прозорецот за декларирање на променливи. Секој контакт, без разлика дали е влез, излез или внатрешна променлива, може да се користи низ целата програма. Врската помеѓу контактите зависи од логиката која сакаме да се реализира: сериска, паралелна или сериска и паралелна.

На еден контакт можат да се придружат само променливи од тип Boolean (BOOL).

Тип на контактот	Симбол
Нормално отворен	—     —
Нормално затворен	—   /   —
Позитивен раб	—   P   —
Негативен раб	—   N   —
Позитивен и негативен раб	—   PN   —

Кај контактите често се среќаваат поимите нормално отворен и нормално затворен контакт. Нормално отворениот контакт (на пример тастер) ќе спроведува електрична

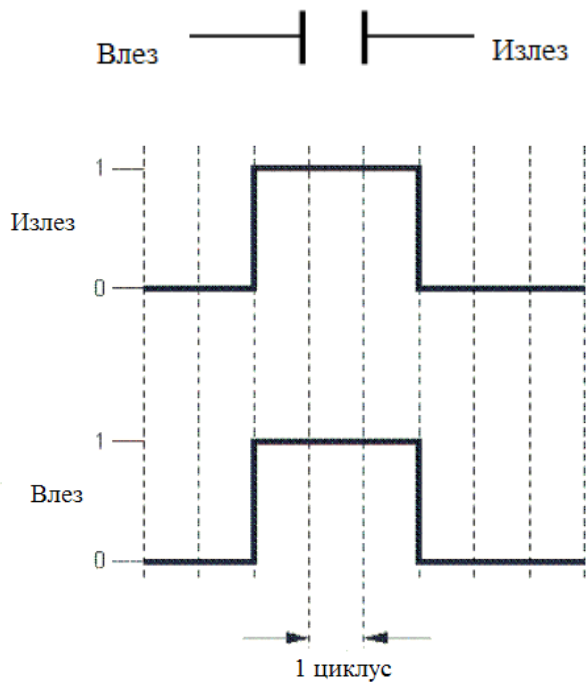
струја, ако е притиснат. Ако се отпушти тастерот, нормално отворениот контакт ќе престане да спроведува струја. Кај нормално затворениот контакт (на пример свонче) е обратно – тој ќе спроведува струја (свончето ќе свони), се додека не се притисне прекинувачот на свончето, со кој ќе се прекине струјното коло.



Слика 4-3 – Нормално отворен и нормално затворен контакт

Пример за употреба на нормално отворениот контакт во пракса е кај заштитните врати на машините. Ако се отвори вратата, контактите се раздвојуваат и струјното коло се прекинува. Нормално отворени и нормално затворени контакти можат да се изведат на излезите на сензорите.

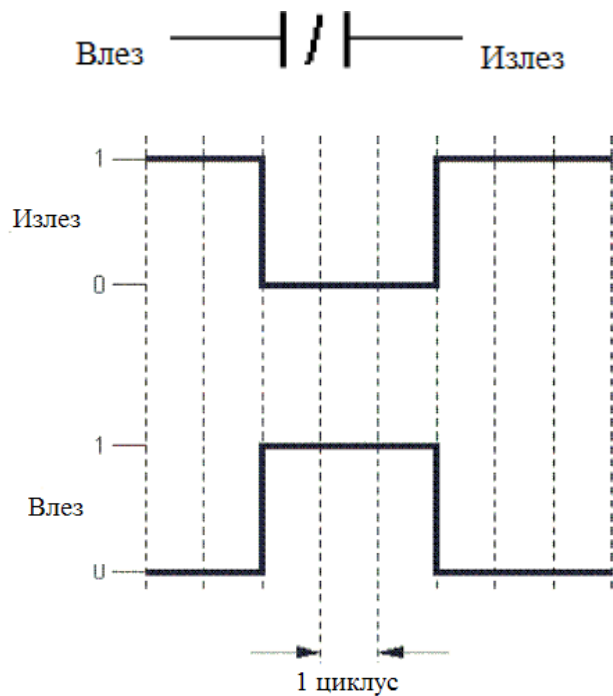
### 1. Нормално отворен контакт



Ако контактот не е притиснат, електричното коло не е затворено и логичката состојба е 0 (False).

Ако контактот е притиснат, физичката состојба преминува во логичка 1 (True).

## 2. Нормално затворен контакт

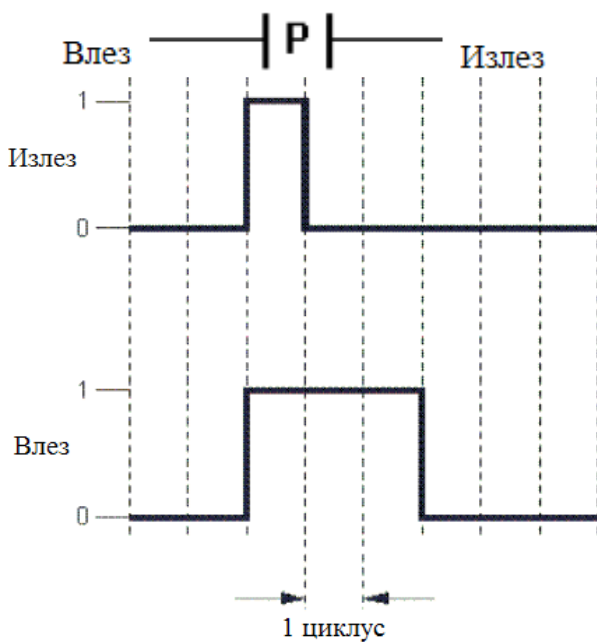


Овој симбол го менува статусот на променливата од тип Boolean.

Се користи на оние места каде што влезниот сигнал не треба да биде присутен за да се изврши инструкцијата на излезот.

Состојбата на излезот е на логичка 0 ако влезот е на логичка 1.

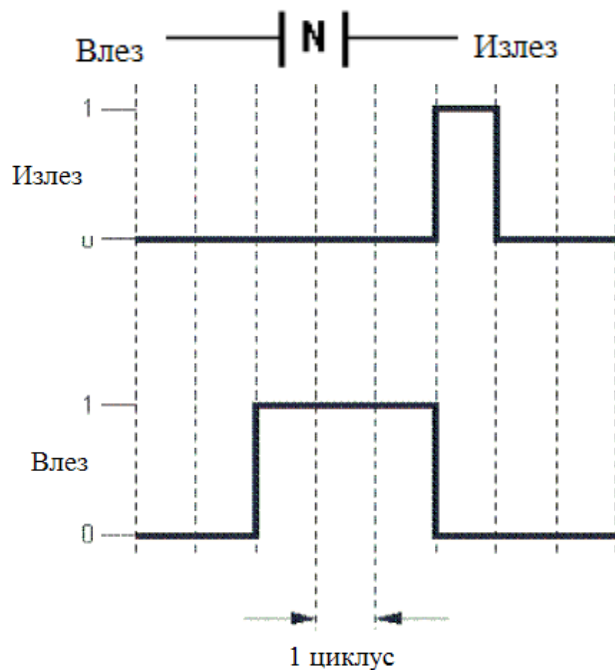
## 3. Позитивен раб



Овој симбол се користи за да формира позитивен раб на дигиталниот сигнал.

Кога вредноста на променливата ќе премине од 0 на 1, се појавува позитивен раб и овој контакт враќа вредност 1 за еден циклус. Ова може да се искористи за да сетира или ресетира променливи или да ги брои позитивните рабови на таа променлива.

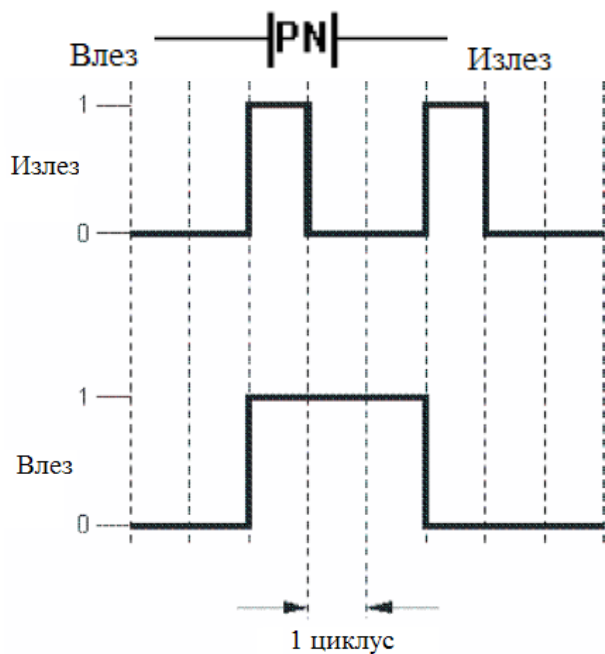
#### 4. Негативен раб



Овој симбол се користи за да формира негативен раб на дигиталниот сигнал.

Ако вредноста на променливата премине од 1 на 0, се појавува негативен раб и овој контакт враќа вредност 1 за еден циклус. И ова може да се искористи за да сетира или ресетира променливи или да ги брои негативните рабови на таа променлива.

#### 5. Позитивен и негативен раб



Овој симбол се користи за да формира позитивен и негативен раб на дигиталниот сигнал.

Однесувањето на овој контакт е еднаков на паралелно поврзани контакти со позитивен и негативен раб. Кога променливата ќе премине од 0 на 1 (позитивен раб) излезот од контактот е 1 за еден циклус. Истото се случува и при премин од 1 на 0 (негативниот раб).

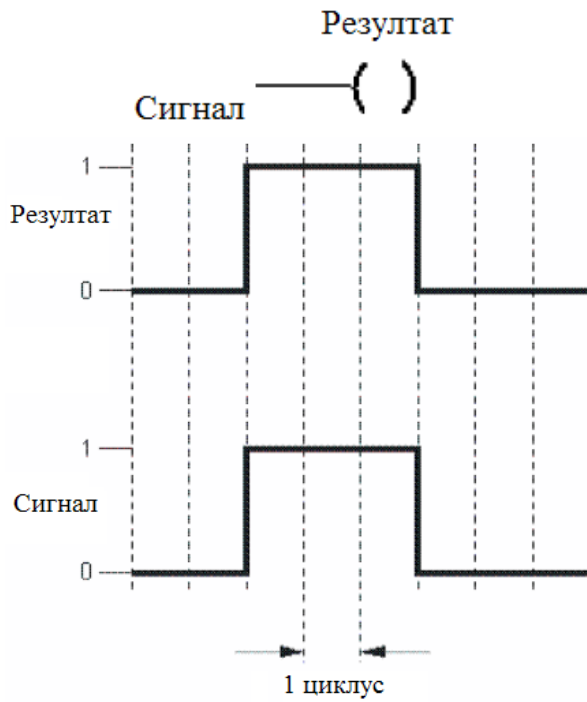
### 7.2.4.2. Намотки

Намотката е еден од основните елементи на ледер дијаграмот. Секогаш е поставена на десната страна од ледер дијаграмот како излез. Намотките се поврзуваат на десната страна од контактите или на десната страна на излезите од функциските блокови. Во еден ледер дијаграм мора да биде присутна најмалку една намотка. На една инструкциска линија (на десната страна на условот) може да се поврзат неколку намотки (излези) поврзани паралелно.

Секоја намотка може да се искористи како дигитален излез или внатрешна променлива, која подоцна може да се искористи како влез во некоја друга мрежа. На намотките можат да им се доделат само променливи од типот Boolean.

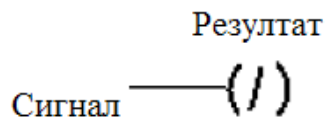
Тип на контактот	Симбол
Намотка	—( )
Негирана намотка	—(/)
Намотка за сетирање	—(S)
Намотка за ресетирање	—(R)
Намотка со премин на позитивен раб	—(P)
Намотка со премин на негативен раб	—(N)
Намотка со премин на двата раба	—(PN)

## 1. Намотка

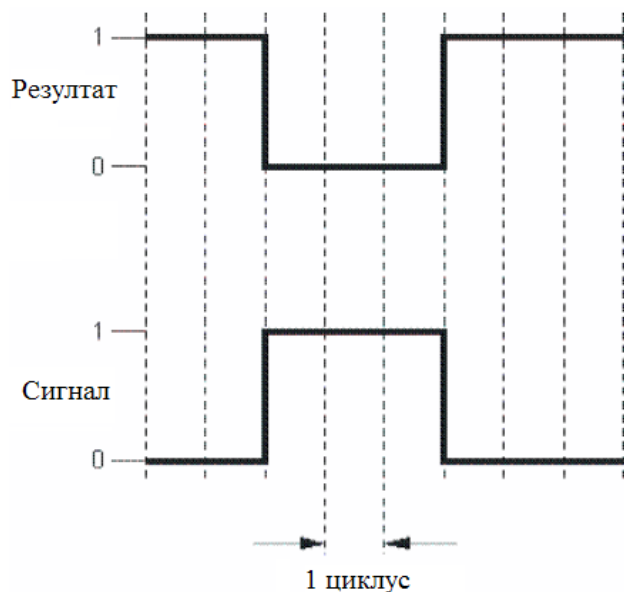


Намотката има состојба логичка 1 ако се исполнети условите во инструкцискиот дел.

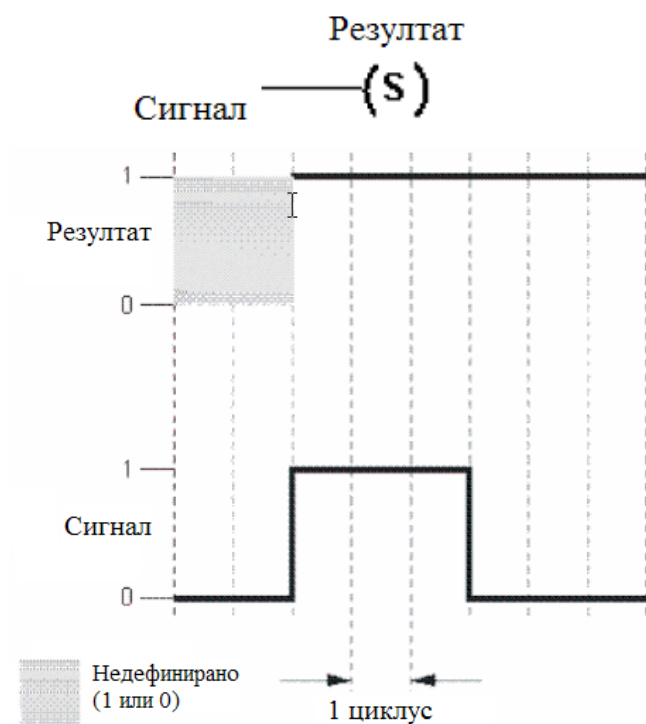
## 2. Негирана намотка



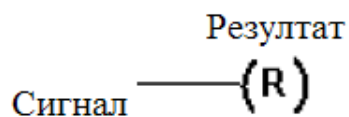
Ако се исполнети условите, негираната намотка има состојба на логичка 0, во спротивно, има состојба на логичка 1.



### 3. Намотка за сетирање



### 4. Намотка за ресетирање

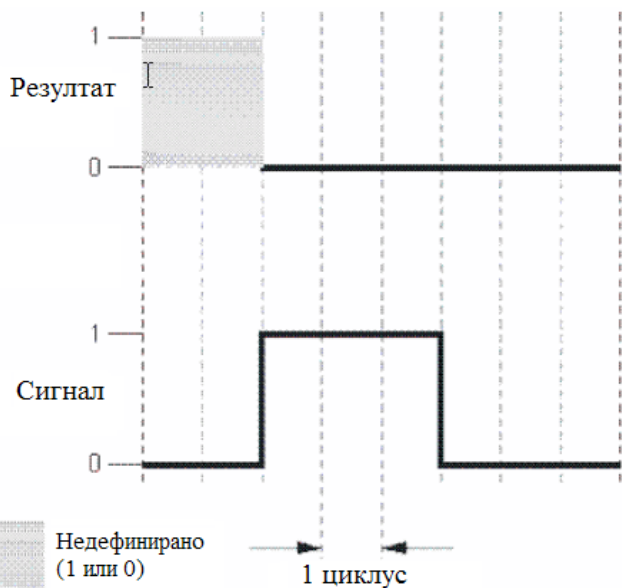


Намотката за сетирање ја сетира променливата во состојба 1 ако се исполнети условите.

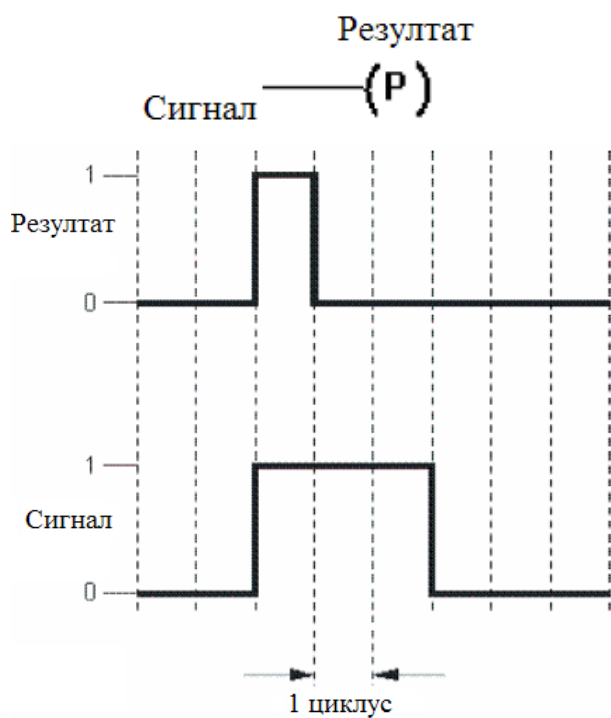
Состојбата останува сетирана се додека променливата не се ресетира.

Ако се исполнети условите, намотката за ресетирање ја ресетира променливата, т.е. ја променува нејзината состојба во 0.

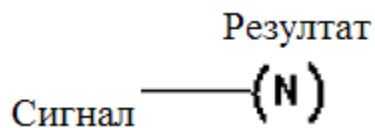




5. Намотка со премин на позитивен раб



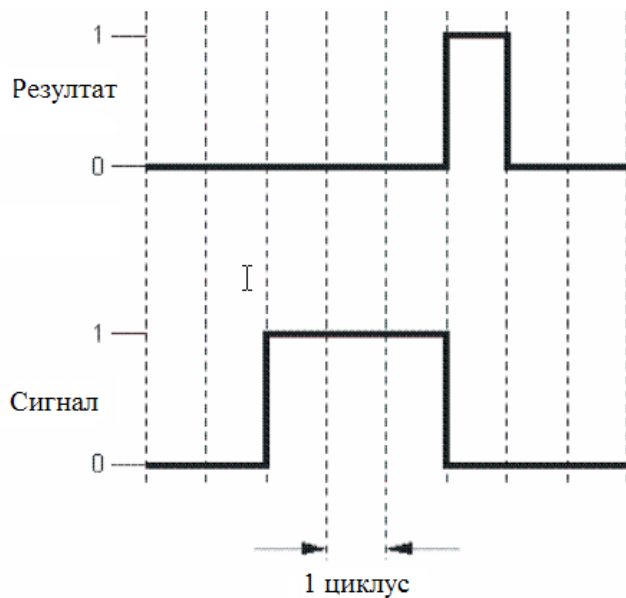
6. Намотка со премин на негативен раб



Намотката со премин на позитивен раб ја променува состојбата на променливата во 1 (во времетраење од 1 циклус) ако се исполнети условите.

За сите други циклуси при исполнети услови, излезот останува со состојба 0.

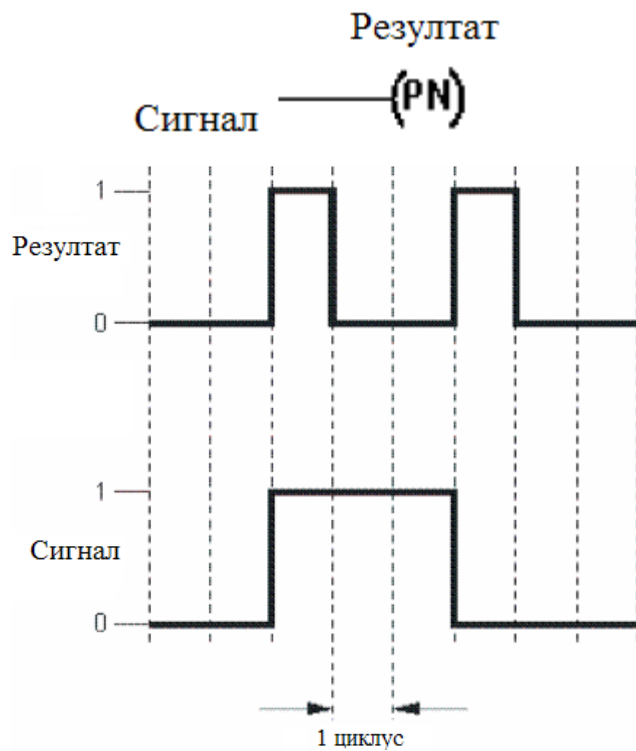
Намотката со негативен раб ја променува состојбата на променливата во логичка 1 за времетраење од 1 циклус, ако не се



исполнети условите, и тоа во првиот циклус по престанокот на исполнувањето на условите (на негативниот раб).

За сите други циклуси во кои не се исполнети условите, вредноста на променливата останува 0.

### 7. Намотка со премин на позитивен и негативен раб



Оваа намотка ги обединува функциите на излезите со позитивен и негативен раб.

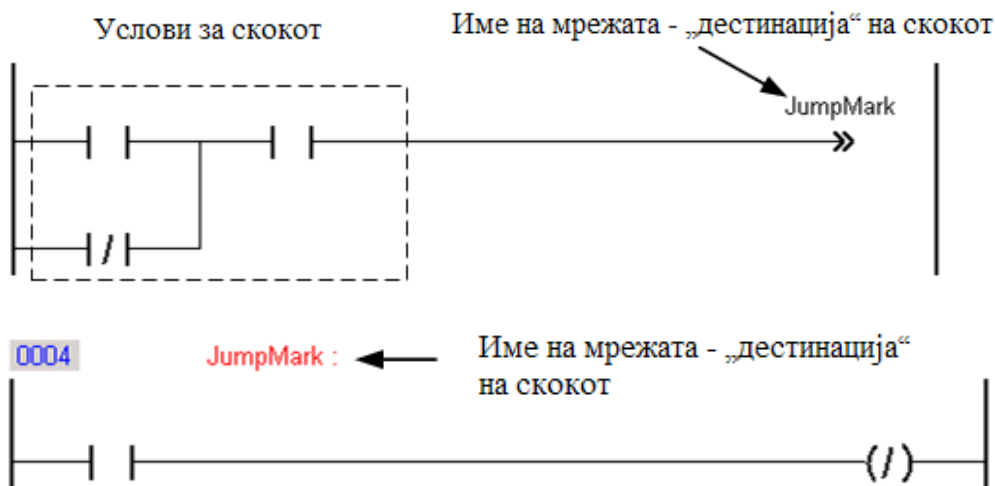
### 7.2.5. Контролирање на текот на програмата

Текот на програмата се контролира со помош на две инструкции, **jump** и **return**. Идејата е оние мрежи од програмата кои не се потребни во дадениот момент да се прескокнат и да се изврши некој дел од програмата што се наоѓа под тие редови.

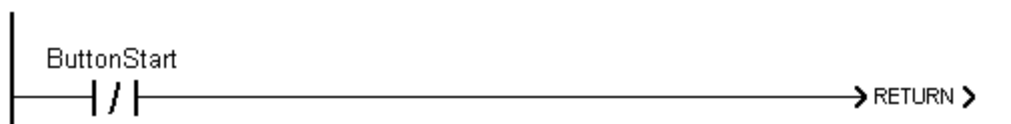
Кога се исполнети условите за инструкцијата jump, таа се извршува на тој начин што извршувањето на програмата прави „скок“ од местото на инструкцијата jump до онаа мрежа со специфично име, т.е. името на инструкцијата jump. На пример, ако инструкцијата jump е означена како „Jump Mark“, ќе се прескокнат сите мрежи после инструкцијата и програмта ќе продолжи да се извршува од мрежата што го носи името „Jump Mark“. За сите скокови потребни се единствени имиња (слика 7.3).

Со ова се овозможува ефикасна контрола на текот на програмата. Ова помага да се скрати времето на извршување на програмата, со елиминирање на оние мрежи кои нема потреба да се извршат во дадениот момент, т.е. при зададените услови.

Инструкцијата return се користи за да го заврши ледер дијаграмот на одредена мрежа, различна од последната, ако се исполнети условите кои се поставени. Ако се исполнети условите на инструкцијата return, оние мрежи што се наоѓаат под неа не се извршуваат (слика 7.4).



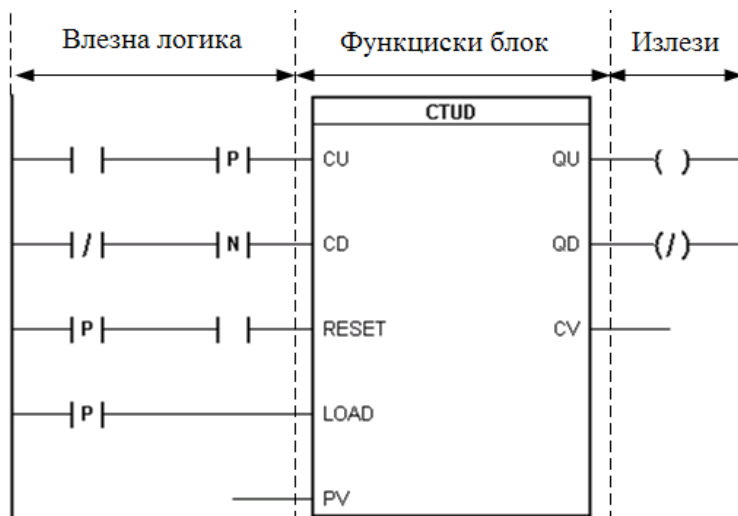
Слика 7.3 – Скок од една мрежа до друга во програмата (инструкција jump)




Слика 7.4 – Инструкција return

### 7.2.6. Користење на функционални блокови

Ледер дијаграм едиторот во Automation Studio овозможува користење на функционални блокови. Ако се внесе функционален блок, тогаш влезната логика (условите) се исто така претставени со инструкции со контакти. Тие ја одредуваат логиката за функционалните блокови. Еден функционален блок може да има еден или повеќе намотки како излези, во кои се регистрира статусот или резултатот на функцијата. Ако функционалниот блок треба да биде активен цело време, тогаш тој се поврзува за вертикалната линија на левата страна (bus bar линија).



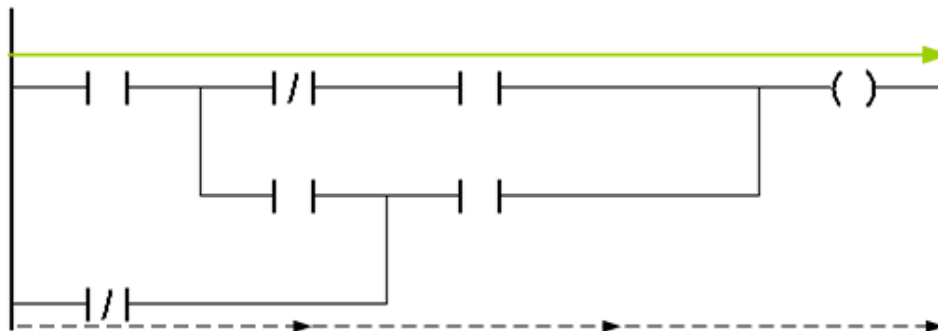
Слика 7.5 – Функциски блок во ледер дијаграм

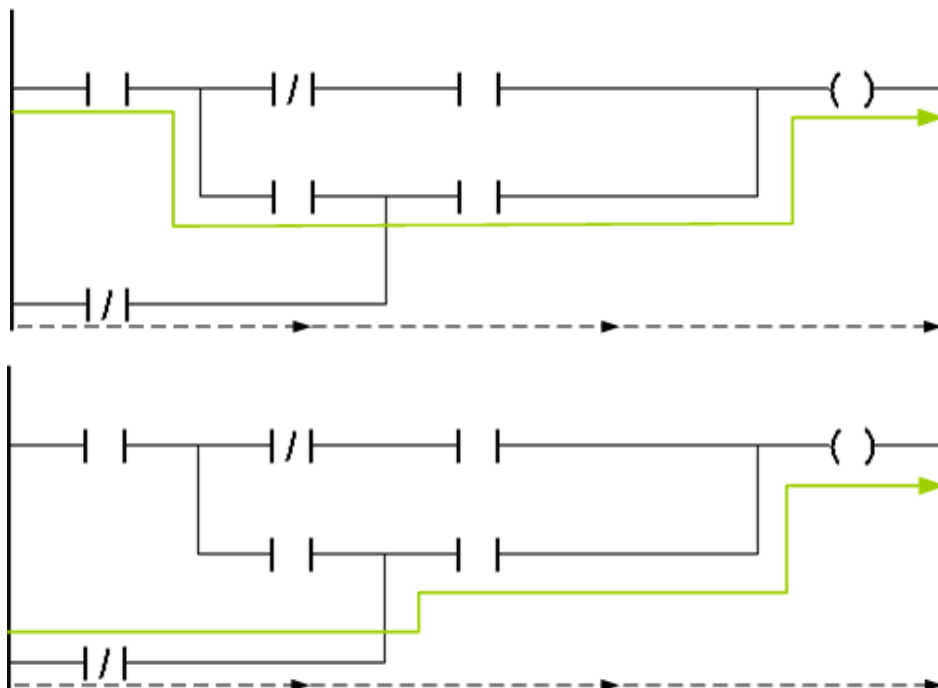
Функцискиот блок може да има и аналогни влезови и излези, поврзувајќи го со променливи на кои им се придружени некои од аналогните влезови и излези на модулите. Потребно е да курсерот да се постави во позиција на некоја од променливите на функцискиот блок (на пример, променливата CV од слика 7.5), и да се притисне иконата  (Analog value) или spacebar-от на тастатурата. Потоа на таа функциска променлива и се доделува некоја од програмските променливи, претходно декларирани.

### 7.2.7. Тек на „струјата“ во програмата

Ако се појави логички континуитет во една мрежа, тогаш состојбата на излезот има состојба логичка 1, т.е. „струјата протекнува“ до делот од мрежата резервиран за излезните инструкции. „Струјата тече“ од лево кон десно во една мрежа. Мрежите се извршуваат една по друга, освен ако текот се промени со користење на скокови или прекинит (инструкциите jump и return).

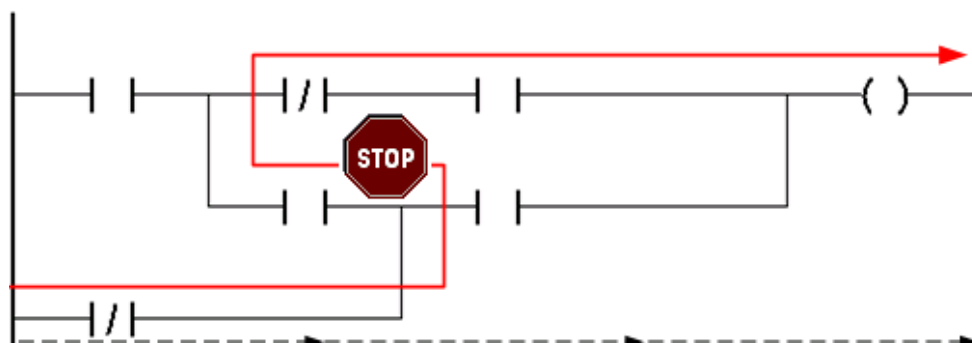
Во следнава мрежа има неколку можни начини на тек на логичкиот континуитет.





Слика 7.6 – Можни начини на тек на логичкиот континуитет

За разлика од жичано поврзаната релејна логика, тек на „струјата“ како што е прикажан на слика 7.7 не е возможна кај логиката на PLC контролерите.



Слика 7.7 – Не евозможен тек на логички континуитет од десно на лево

### 7.3. Променливи

Променливите се симболички елементи кои се користат во програмирањето. Тие претставуваат мемориски локации од кои што можат да се читаат и запишуваат податоци, со пристапувањето до променливата. Со користењето на овие симболички елементи е овозможено корисникот да не води многу сметка за користењето на меморијата, бидејќи тоа го управува задачата на програмирањето.

Константите се вид на „променливи“, чијашто вредност не се менува. Таа е зададена за време на креирањето на софтверот и нејзината вредност може да биде само прочитана.

### 7.3.1. Типови на податоци

Типот на податоците ги опишуваат својствата на променливите. Примери за својства можат да бидат можниот опсег на бројот складиран во променливата, неговата прецизност, или можните операции кои можат да се извршат врз него.

Следните типови на податоци се викаат основни типови на податоци. Тие можат да се користат во сите програмски јазици.

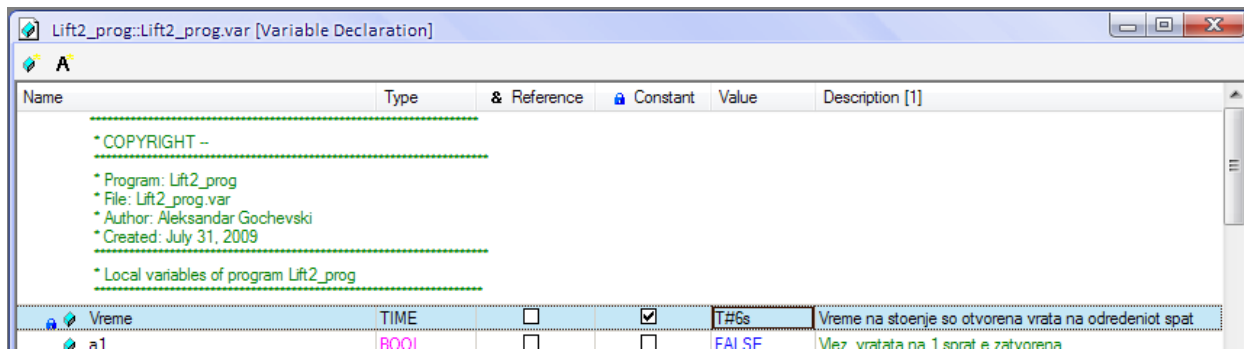
Бинарни	Целобројни ненегативни	Целобројни	Реални	Време, датум, знаковни променливи
BOOL	USINT	SINT	REAL	TIME
	UINT	INT	LREAL	DATE_AND_TIME
	UDINT	DINT		STRING

Тип на податок	Потребна меморија (бајти)	Опсег
BOOL	1	TRUE (1), FALSE (0) Дигитални влезови и излези
SINT	1	-128 ... +128
INT	2	-32768 ... +32768 Аналогни влезови и излези
DINT	4	-2147483648 ... +2147483647
USINT	1	0 ... 255
UINT	2	0 ... 65535
UDINT	4	0 ... 4294967295
REAL	4	-3.4E38 ... +3.4E38
LREAL	8	-1.79769313486231E308 ... +1.79769313486231E308
TIME	4	T#-24d_20h_31m_23s_648ms ...T#24d_20h_31m_23s_647ms
DATE_AND_TIME	4	DT#1970-01-01-00:00:00 ... DT#2106-02-07-06:28:15
STRING	Променлива	Приказ на знакови


### 7.3.2. Декларирање на променливи и константи

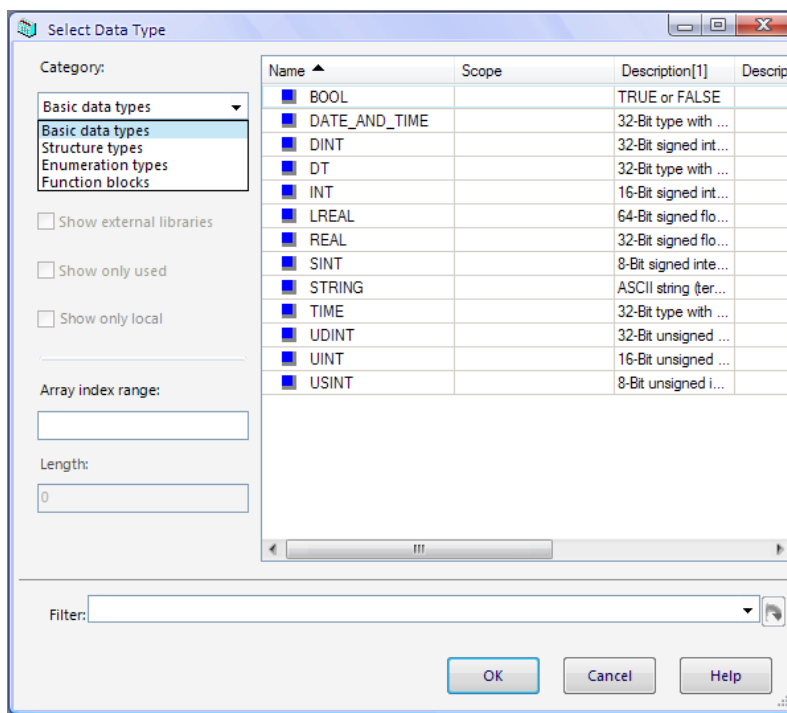
Променливите и константите се декларираат во фајлот со додавката (екстензија) **\*.var**. Начинот на декларирање на променливите беше објаснет во точката 3.2.2. При деларирањето на променливата, освен нејзиното име, треба да се специфицира и типот на

податокот што таа ќе го носи и дали таа е константа. На една променлива или константа може да се специфицира и одредена вредност (слика 7.8).



Слика 7.8 – Декларирање на константа и специфицирање на вредносот на променливата (константата)

Во колоната **Type** (слика 7.8) се специфицира типот на променливата, односно типот на податоците кои можат да се доделат кон променливата. Најпрво, се кликува (се селектира) променливата и се кликува во редот на името на променливата и колоната **Type**. Откако ќе се појави иконата  се кликува на неа и се отвора прозорец како на слика 7.9.



Слика 7.9 – Избор на типот на променливата

Може да се избере помеѓу основни типови на променливи (Basic data types), структурирани типови на променливи (Structure types), набројни типови на податоци (Enumeration types) и типови на податоци на функциските блокови. Последниот тип на податоци се доделува на променливите што се доделуваат на функциските блокови. На

пример, ако на функцискиот блок TON е доделена променливата Tajmer1, таа треба да биде од типот TON, кој може да се најде ако во паѓачкото мени **Category**, (слика 7.9), се избере Function blocks.

### 7.3.3. Структури (кориснички типови на податоци)

Корисникот (програмерот) може да групира одредена група на променливи во структура. Ова овозможува да одделните променливи кои би биле „расфрлани“ наоколу, декларирани со други типови, да се групираат и да формираат структури кои ќе рефлектираат одредена функција или задача.

Ова подобро може да се објасни со пример: Нека е зададена задача да се креира програма која ќе управува со печење на два типа на леб. Секој тип на леб се дефинира со променливите Voda, Brasno, Sol и Kvasec. Типот на податоците ќе ги содржи следните елементи:

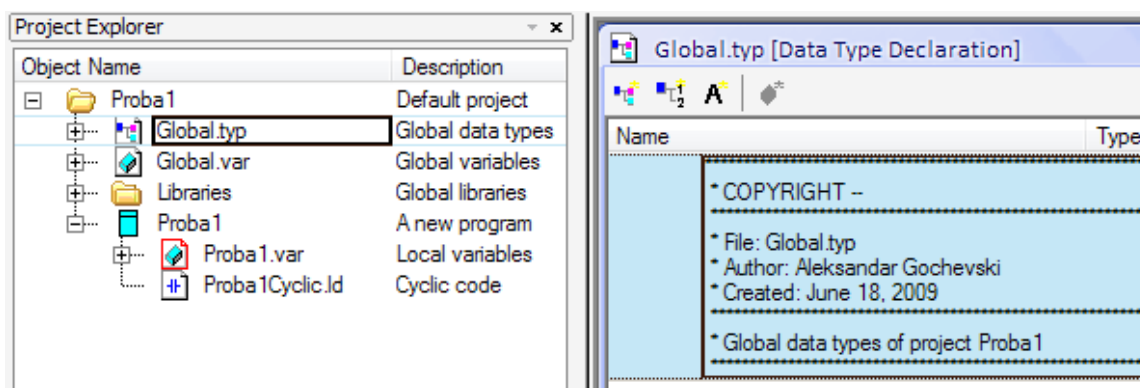
Voda  
Brasno  
Sol  
Kvasec

Типовите на лебови се мешан леб и домашен леб. Предноста на структурата е во тоа што се потребни само две променливи, на пример “mesan\_leb” и “domasen\_leb”. И двете променливи ги содржат потребните елементи (вода, брашно, сол и квасец).

Ако би требало да се прошири програмата со уште еден тип на леб, она што треба да се направи е само да се креира уште една променлива од типот rescept\_na\_leb (на пример, “bel\_leb”). Ако подоцна има потреба да се специфицира и времето на печење, тоа се прави со едноставно проширување на структурата, со елементот Vreme\_na\_resenje.

Во овој пример, со помош на структурата, се користат само 3 променливи од типот rescept\_na\_leb, наместо 15 кога не би постоела структурата.

За да се креираат типови на податоци, треба да се отвори фајлот со екстензија \*.typ. Тоа е Global.typ (слика 7.10), се отвора со двоен клик и неговиот прозорец се појавува на десната страна од главниот прозорец.

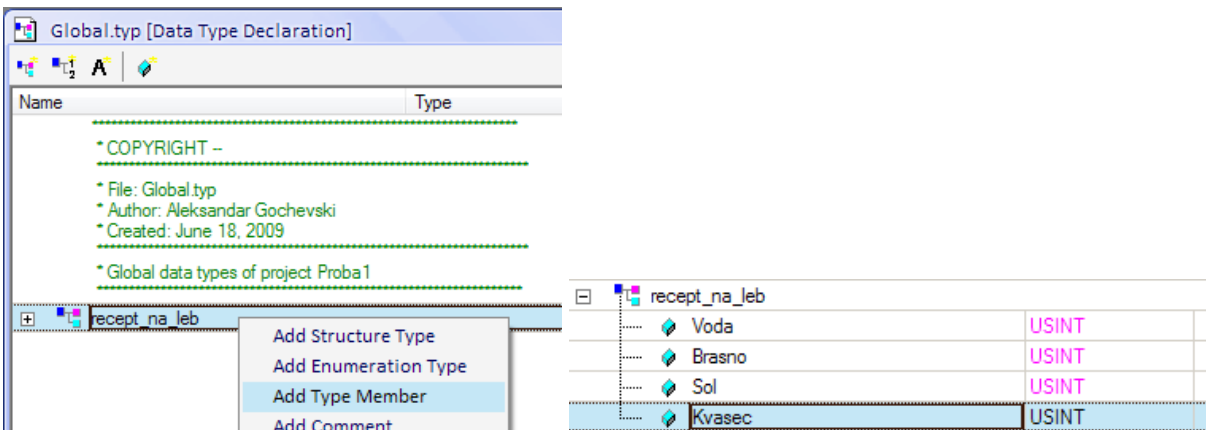


Слика 7.10 – Отворање на декларацијата за типови на податоци

Со десен клик на новоотворениот прозорец и со избор на **Add Structure Type** се декларира нов тип на податок, на кој му даваме име (пример, rescept\_na\_leb). Содржината



на типот на податоците, елементите, се додаваат со избор на **Add Type Member** од менито кое се отвора со десен клик на името на структурата (слика 7.11).

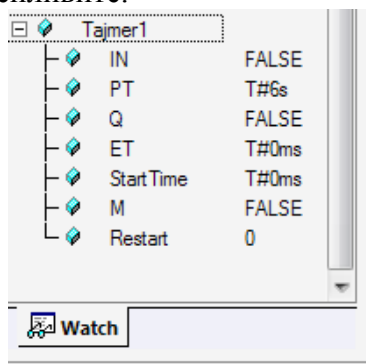


Слика 7.11

Откако ќе се зачува, овој тип на податоци може веднаш да се користи во програмите од проектот.

### 7.3.4. Типови на податоци на функциските блокови

Секој функциски блок има влезови и излези, заедно групирани во форма на структура. Кога еден функциски блок ќе се повика, програмата „позади“ функцискиот блок ја прима таа структура на податоци. Ако во прозорецот за следење на вредностите на променливите (Watch прозорецот) се додаде една променлива од типот на функцискиот блок на кој е придружена, ќе се види дека еден функцискиот блок е составен од одделни елементи, кои со текот на извршувањето на програмата на функцискиот блок, во општ случај се менуваат. На слика 7.12 е даден пример на функцискиот блок TON поставен во прозорецот за следење на променливите.



Слика 7.12 – Променливата Tjajmer1 на функцискиот блок TON во Watch прозорец

### 7.3.5. Низи

Низите се променливи што содржат неколку елементи од ист тип на податоци. Кон тие елементи се пристапува со помош на индекс. Овие елементи можат да се декларираат како основни типови на податоци (проста низа) или како кориснички типови на податоци (низа од структури). Индексот на низата секогаш започнува со 0.

Пристапот до елемент од низата изгледа вака:

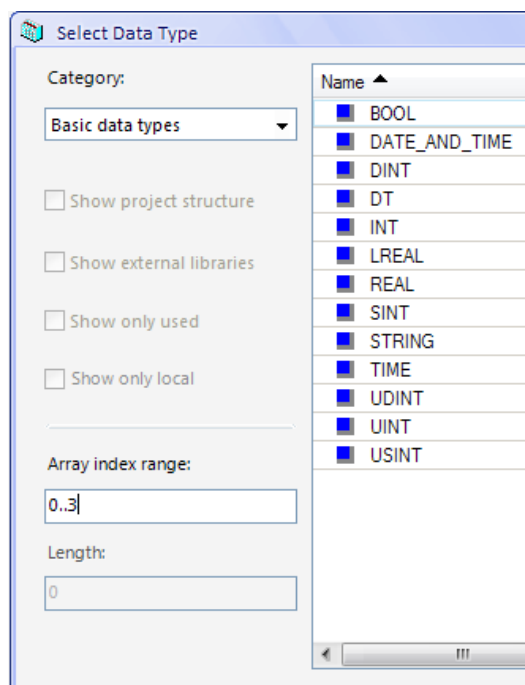
```
ArrayVariable[ArrayIndex]
```

Пристапот до низа од структура изгледа вака:

```
ArrayVariable[ArrayIndex].Element
```

Во Automation Studio променливата се декларира како низа во прозорецот за декларирање на променливите, при изборот на типот на податокот.

Низите од податоци се користат кога се потребни променливи од ист тип на податоци.



Слика 7.13 – Декларирање на низа

### 7.3.6. Домен на променливи

Пакетите на еден проект можат длабоко да се вгнездуваат (структурираат), во зависност од потребата, а тоа може да се види во логичкиот поглед. Ова овозможува енкапсулација (обвивање) на податоците. Структурата на проектот го определува доменот (видливоста) на користење на декларираните променливи и типови на податоци. Постојат разлики во доменот на променливите:

- Глобални променливи, се наоѓаат на највисокото ниво и се видливи за целиот проект. Тие исто така се глобални и од гледна точка на контролерот.

- Локални променливи на пакетот. Тие се декларираат во границите на пакетот и можат да се користат во сите под-пакети или програми. И овие променливи се глобални од гледна точка на контролерот.
- Локални променливи, се декларираат во програмата и се видливи само за конкретната програма. Од гледна точка на контролерот, тие се локални променливи.

Object Name	Description
Machine	
Global.typ	Global data types
Global.var	Global variables
Libraries	Global libraries
Part_A	A package of related programs and data objects.
Part_A.typ	Global data types
Part_A.var	Global variables
SupPart_A	An empty package
ProgX	A program in IEC-1131 languages, B&R Automation Basis or ANSI-C
ProgX.var	Local variables
ProgXCyclic.ld	Cyclic code
Documentation_A.pdf	
PartB	A package of related programs and data objects.
PartB.typ	Global data types
PartB.var	Global variables
MyProg1	A program in IEC-1131 languages, B&R Automation Basis or ANSI-C
Documentation_B.pdf	
Steps	A program in IEC-1131 languages, B&R Automation Basis or ANSI-C
Steps.var	Local variables
StepsCyclic.ab	Cyclic code
_CYCLIC	Coment

Слика 7.14 – Глобални и локални променливи

### 7.3.7. Иницијализација на променливите и константите

Променливите треба да имаат дефинирани вредности во секој момент. Има неколку начини на иницијализација на променливите: од системот или од корисникот. Иницијализацијата се одвива по следниот редослед:

- Во прозорецот за декларирање на променливи
- Во иницијализацијата на задачата
- Во делот на цикличната задача

**Декларирање на променливата.** Вредноста на иницијализација (почетната вредност) може да се внесе за променливите и константите во прозорецот за декларирање на променливите, и тоа во колоната **Value** (слика 7.8). Притоа постојат две можности:

- Променливите да се иницијализираат со **фиксна** вредност (нумеричка вредност што припаѓа во опсегот на вредности на променливата)

- Променливите да се дефинираат како **заостанати**. Овие вредности се зачувани во областа на бафер меморијата (меморијата напојувана од батерија), пред рестартирањето на системот. Една променлива се декларира како заостаната, ако во колоната **Value**, од паѓачкото мени се одбере RETAIN (слика 7.15).

 condition	USINT	<input type="checkbox"/>	<input type="checkbox"/>	RETAIN
 setValue	USINT	<input type="checkbox"/>	<input type="checkbox"/>	0

Слика 7.15

**Иницијализација на задача** се случува (ако постои), така што секој циклус на задачата преку својата потпрограма за иницијализација, кога цикличниот систем се стартува (ова се случува пред да се стартува цикличниот дел од програмата). Програмата за иницијализација може да содржи програмски код што ги дефинира вредностите на променливите.

**Иницијализација во текот на цикличната задача.** Цикличниот дел на програмата започнува после декларирањето на променливите и иницијализацијата на задачата. Оние променливи на кои им се доделени вредности, ги задржуваат се додека не примат нови или системот не се рестартира.

**Заостанати променливи.** Заостанатите променливи се зачувуваат во посебна безбедна меморија за време на рестартирањето на системот (топол рестарт или снемување на струја), од каде што можат повторно да бидат прочитани, откако системот ќе заврши со рестартот. Податоците се зачувуваат благодарение на дополнителното напојување (батеријата) на процесорот.

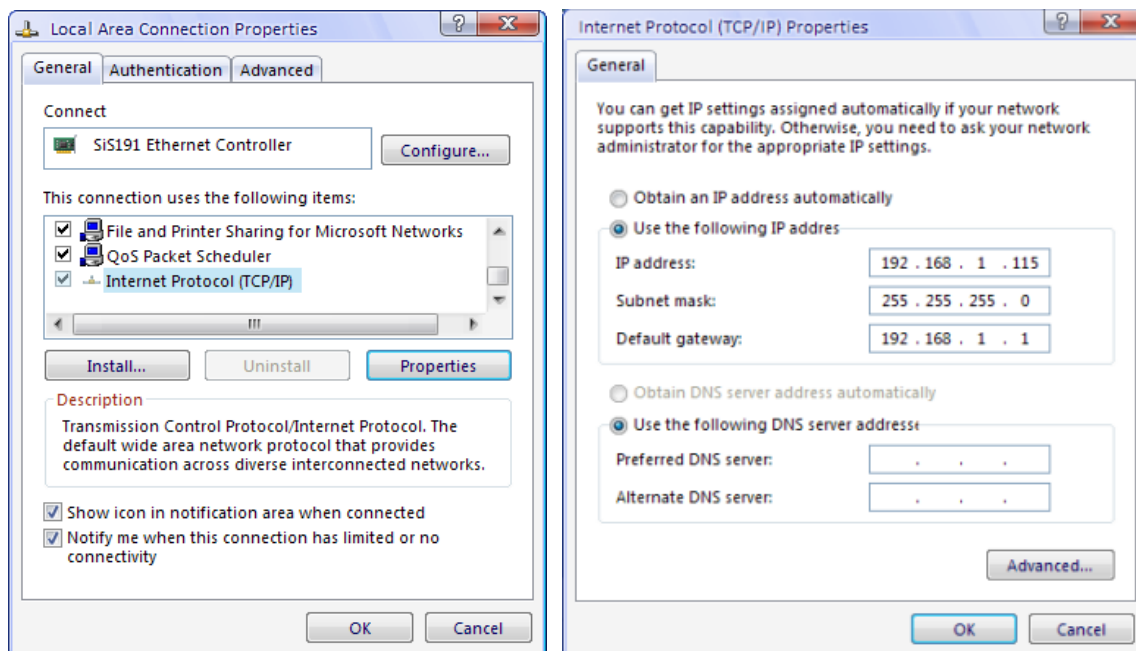
## 8. Практично извршување на апликација во лабораториски услови

Во ова поглавије ќе бидат презентирани подесувањата потребни за воспоставување врска на компјутерот со контролерот, поврзувањето на сензорите и актуаторите со контролерот, однесувањето на процесот, како и резултатите што ги дава симулацијата.

### 8.1. Потребни подесувања за воспоставување на врска помеѓу компјутерот и контролерот и пренос на проектот во контролерот

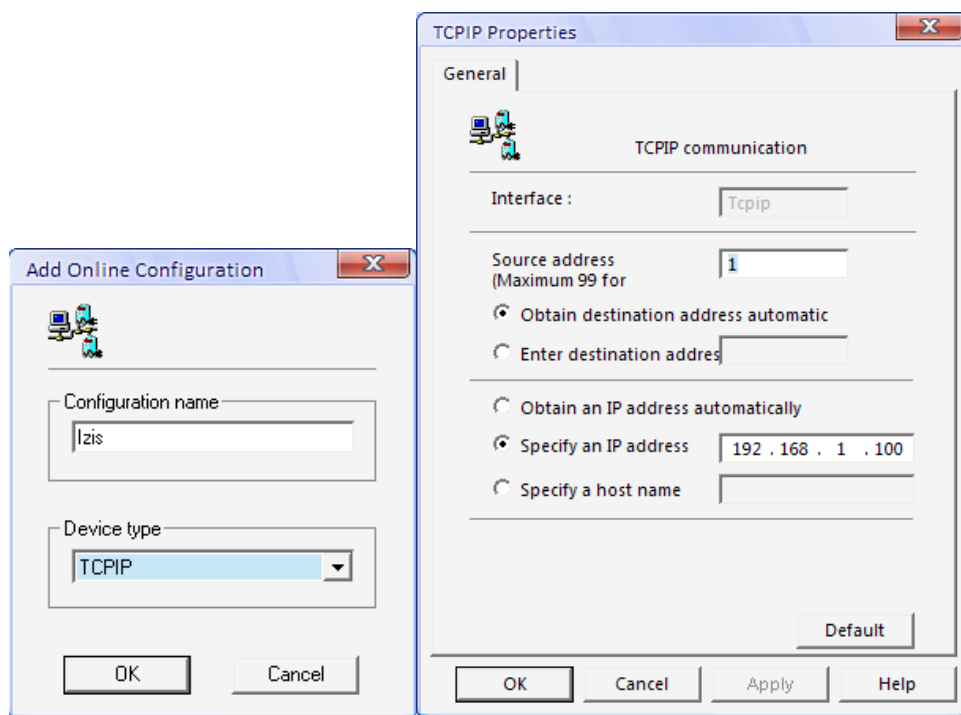
Ова е првата работа што треба да се направи, да се пренесе проектот од персоналниот компјутер, каде што е креиран, до целниот систем, програмибилниот контролер. Воспоставувањето на оваа врска е значајна и за мониторинг и интервенции на состојбите на променливите за време на тестирањето на управувачкиот софтвер. Проектот може да се пренесе на контролерот и со помош на CompactFlash мемориската картичка. Во овој случај преносот ќе го вршиме со помош на Ethernet локална мрежа. Ќе биде искористена локалната мрежа на факултетот, од каде со помош на рутер се поврзани неколку компјутери, но и програмибилниот контролер (во портот IF2).

Најпрво треба да се направат подесувања на компјутерот за тој да се поврзе со локалната мрежа. Се отвора прозорецот **Network Connections (Start / Connect To / Show All Connections)** и на локалната конекција (**Local Area Connection**) се кликува со десен клик и се одбира **Properties**. Потоа, во јазичето General се одбира категоријата **Internet Protocol (TCP/IP)** и се кликува **Properties** (слика 8.1 – лево). Ќе се отвори прозорец како на слика 8.1 – десно и таму треба да се направат подесувањата како што се прикажани на сликата.



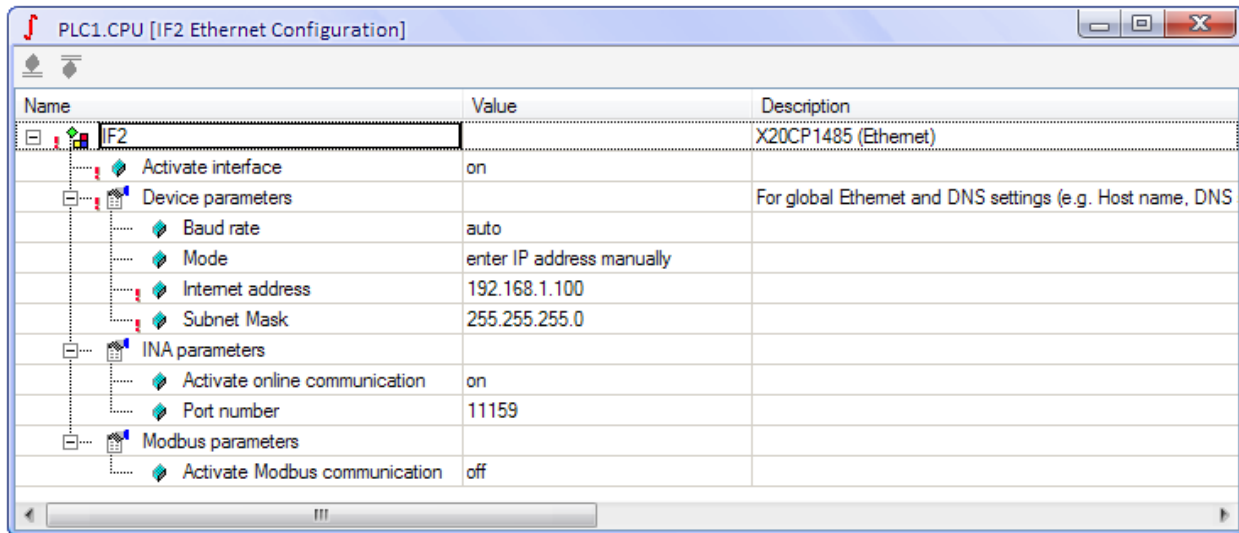
Слика 8.1 – Подесувања за пристап на локалната мрежа

После овие подесувања (мрежниот кабел секако треба да е поврзан во компјутерот) компјутерот има пристап до мрежата, на која и програмибилниот контролер е поврзан. Сега е потребно да се направат подесувања во Automation Studio. Најпрво во конфигурацискиот поглед треба да биде активна онаа конфигурација која е за реално поврзување на компјутерот со контролерот – во случајов тоа е Konfiguracija1 (значи не треба да биде активна конфигурацијата за симулација). Потоа се отвора **Online / Settings** и се додава нова комуникација (со кликување на копчето **Add** на отворениот прозорец). По ова се појавува прозорец како на слика 8.2 – лево, каде треба да се внесе името на конекцијата и водот на комуникацијата. Името е Izis (според името на локалната мрежа), а видот на комуникацијата е TCP/IP. Откако ќе се внесе ова, се кликува на **Properties** за да се подеси врската. Се појавува прозорец како на слика 8.2 – десно и се прават подесувањата како што е прикажано. На контролерот можат да пристапат и повеќе компјутери, но затоа е потребно кај секој компјутер да има различен број во полето **Source address**.

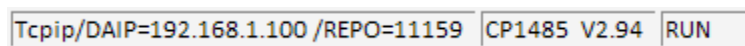


Слика 8.2 – Подесувања за комуникацијата со контролерот

Сега останува уште едно подесување, а тоа се прави кога во физичкиот поглед ќе се отвори Ethernet врската (со десен клик на името на процесорот и **Open Ethernet**). На новоотворениот прозорец на IF2 (Ethernet) се кликува со десен клик и се одбира **Properties**. Потоа ќе се отвори прозорец како на слика 8.3, каде што треба да се направат прикажаните подесувања. Најпрво треба да се активира интерфејсот (**Activate Interface = on**). Потоа, во Mode се одбира **enter IP address manually** и се внесува IP адресата 192.168.1.100 (истата како на слика 8.2 – десно), а во Subnet mask се внесува 255.255.255.0 – исто како на слика 8.1 – десно. Во делот **INA Parameters, Activate online communication** се поставува на **on**, а бројот на портата треба да биде 11159. Се зачувува овој прозорец и комуникацијата треба да се воспостави (слика 8.4).



Слика 8.3



Слика 8.4

Потоа следи класичното префрлување на проектот на целниот систем, со таа разлика што сега тоа не е AR000, туку реалниот управувачки хардвер.

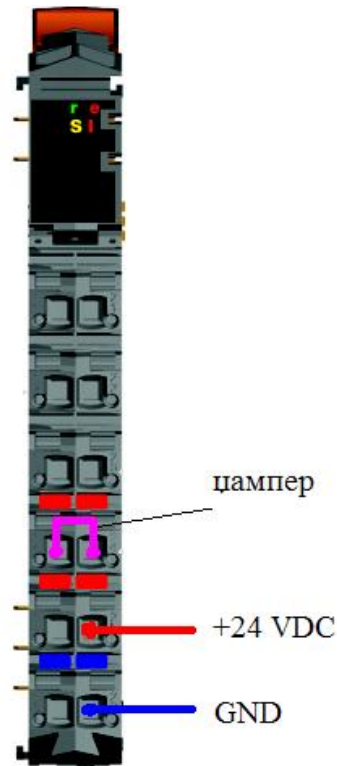
На крајот е добро да се забележи дека погоре претставените подесувања (вредностите на адресите, итн.) важат само за конкретната мрежа која е користена. Во друг случај тие треба да бидат во согласност со мрежата која ќе се користи.

## 8.2. Поврзување на компонентите на симулацијата

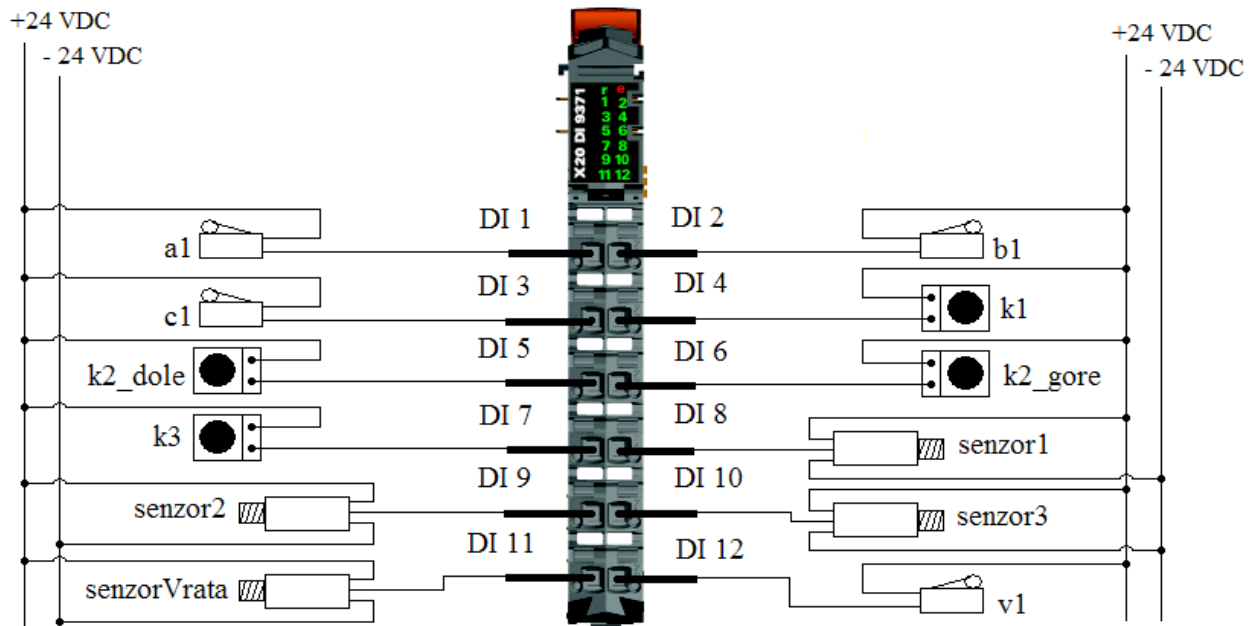
Пред да се изврши поврзувањето на компонентите на симулација, потребно е да се направи шема на поврзувањето. Ова се однесува на поврзувањето на сензорите и актуаторите со контролерот и напојувањето, како и поврзување на пневматските цилиндри со напојување на воздух под притисок. Сите компоненти се поставуваат на посебна табла во лабораторијата. Испитната табла, пневматските цилиндри и распоредните вентили се производство на FESTO и се наменети за изведување на лабораториски испитувања.

На пример, за симулација на работа на лифт со три спрата би можела да се употребат пет пневматски цилиндри, сите двострано управувани: три за вратите на спратовите, и по еден за внатрешната врата на лифтот и цилиндер кој ќе го симулира подигнувањето и спуштањето на кабината. Исто така, потребни се и шест распоредни вентили, и тоа: 4 од типот 5/2 (за отворање и затворање на вратите) и 2 од типот 3/2 (за реализација на подигнувањето и спуштањето на кабината). За регистрирање на крајните положби на извлечените цилиндри на вратите ќе се користат механички крајни прекинувачи, а за одредување на положбата на главниот цилиндер – индуктивни сензори. За регистрирање на присуство на човек, при влегување и излегување од кабината, ќе се користи оптички сензор (би бил поставен на внатрешната врата). Напојувањето со еднонасочна струја е стандардно 24 V, а напојувањето со воздух под притисок е 5 bar.

На наредните слики ќе бидат прикажани шемите на поврзување, каде што на приклучоците се означени имињата на променливите во управувачката програма.

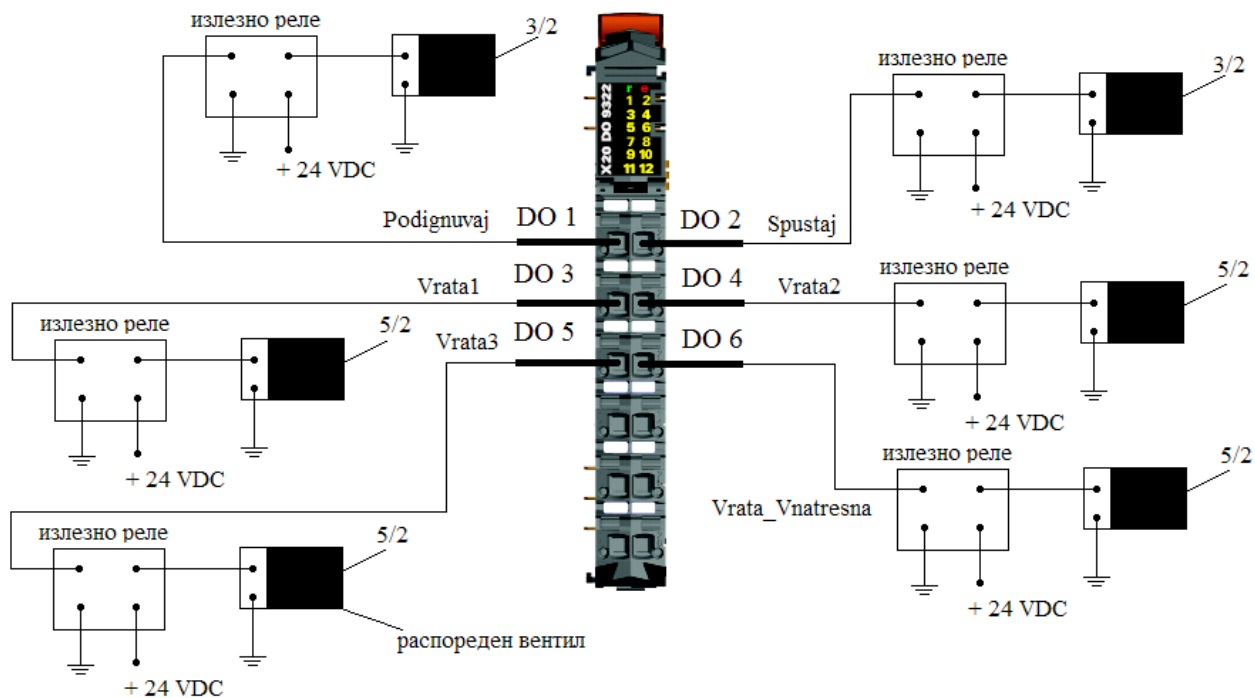


Слика 8.5 – Шема на поврзување на напојувањето



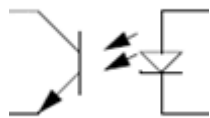
Слика 8.6 – Шема на поврзување на влезите во влезниот модул X20 DI 9371





Слика 8.7 – Шема на поврзување на излезниот модул X20 DO 9322

Во шемата на поврзување на излезниот модул се користат излезни релеа, за да го одвојат колото за напојување на распоредниците од она што е приклучено на излезниот модул. Сигналот од излезните приклучоци го вклучува релето, кое носи напојување за распоредникот, во посебно струјно коло. Со ова се постигнува двојна заштита на електрониката на контролерот. Првата е во самиот модул, каде што сигналот се пренесува со помош на оптокаплер (диода и фототранзистор). Излезните релеа се користат бидејќи постои можност електромагнетите на распоредниците да повлечат поголема струја и со тоа всушност се заштитува оптокаплерот.



Слика 8.8 – Оптокаплер, пренесување на сигналот со помош на светлина

Со погоре прикажаното поврзување и според изработената програма, симулацијата работи како што е предвидено. Следат подесувањата за панелот и реализација на симулацијата со него.

## Додаток 1 – Фотографије од лабораторијата

