

## 1. Програмибилен контролер од типот V&R X20

Во ова поглавје ќе биде опишан контролерот со неговите влезни и излезни модули и операторскиот интерфејс. Контролерот кој ќе се обработува е тип V&R (Bernecker & Rainer) модел X20 CP1485. Исто така ќе се обработат и модулите: X20 DI9371 – дигитален влезен модул, X20 AI4622 – аналоген влезен модул, X20 DO 9322 – дигитален излезен модул и X20 AO 4622 – аналоген излезен модул. Операторскиот интерфејс е Touch Screen дисплеј, исто така производ на V&R, модел 4PP320.0571.35.

## 2. Карактеристики на X20 системите

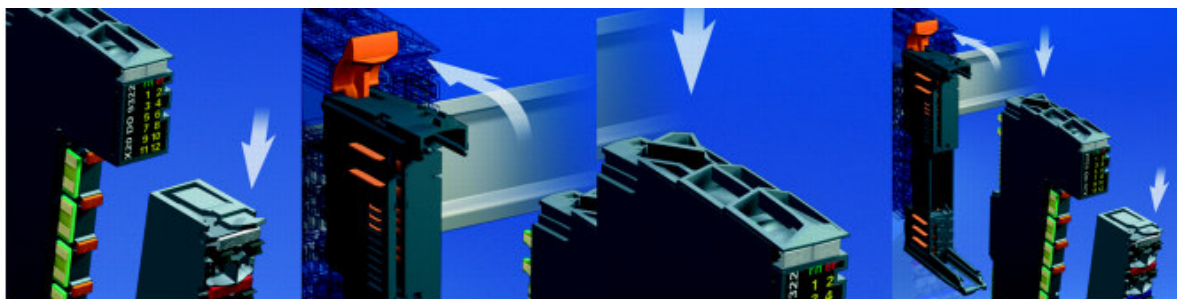
### 2.1.1. Нов стандард во автоматизацијата

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



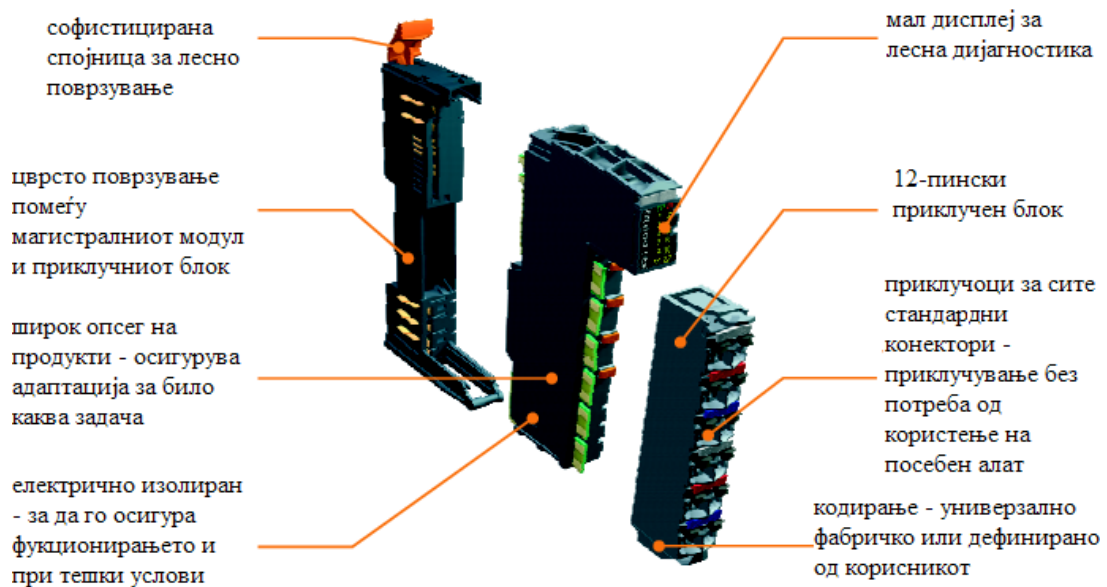
Слика 2.1 – Трите дела на X20 системот

Системот X20 е за 50% подобрен во поглед на интегрираноста и просторната зафатнина на компонентите, усовршена 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 2.2 – Составни делови и особини на еден X20 систем

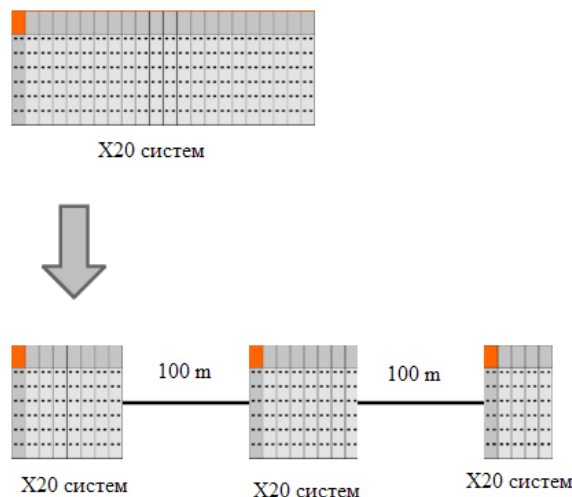
## 2.1.2. Централни процесорски единици (CPU) X20

Новиот, оптимално дизајниран асортиман на X20 процесорски единици задоволува широк опсег на потреби. 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.

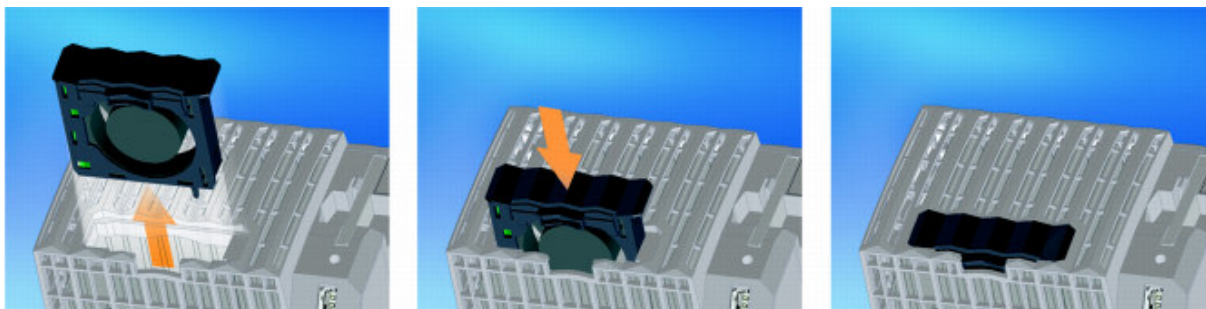
Децентрализираната матична плоча е местото каде можат да се поврзуваат разни модули (влезни, излезни, процесорски итн.) со помош X2X кабел. Тие можат да се постават на растојание и до 100 метри оддалеченост од главниот метален орман. Ова не претставува само обична плоча која е задолжена за комуникација 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 2.3 – 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 2.4).

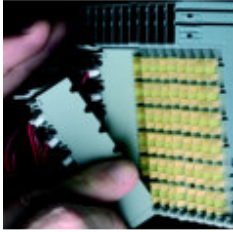


Слика 2.4 – Замена на вентилаторот кај Celeron 650 процесорите

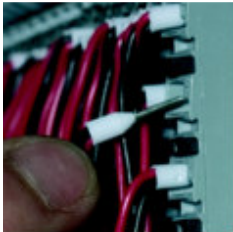
### 2.1.3. Поврзување со приклучоците

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.



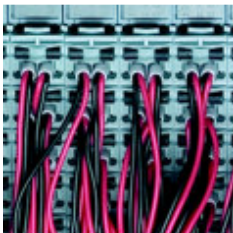
Приклучоците претходно можат да се поврзат одвоено од вистинските влезно – излезни модули. Ова е голема предност во поврзувањето, што обезбедува одвоено производство, навремена логистика и инсталација на претходно склопени системи.



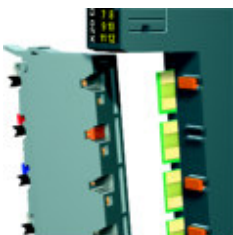
Едноставно и брзо поврзување без користење од посебен алат. Жицата се поврзува со втиснување во приклучокот. Достапни се приклучоци со 6 и 12 пинови.



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.



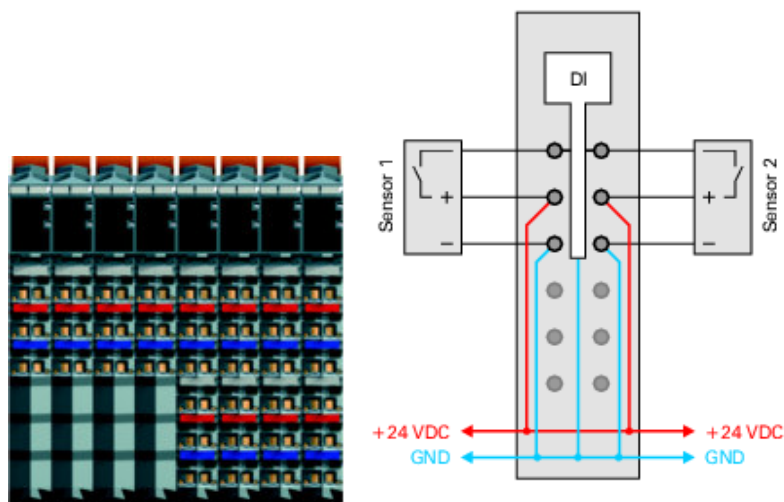
Секој приклучок е јасно означен на пластиката, со што се знае кој приклучок од кој сензор доаѓа или кон кој актуатор води.



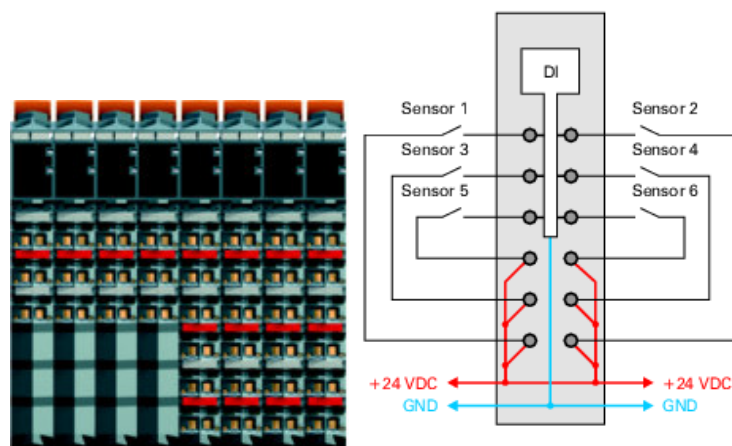


Како додаток на приклучниот конектор и механизмот за отклучување, секој приклучок овозможува пристап за испитување на напонот на приклучокот, без да се одврзува жицата.

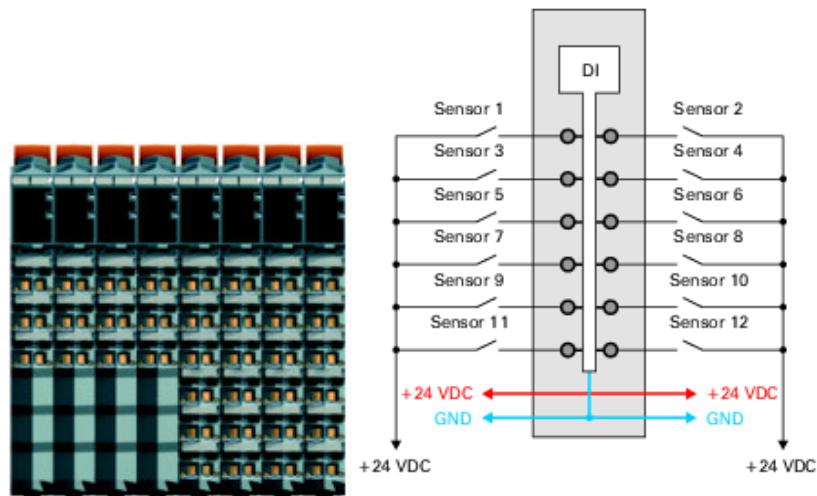
Поврзувањето може да се изведе едножично, двојично или трижично. При тоа не се потребни дополнителни џампери и сите три вида на поврзување може да се комбинираат.



Слика 2.5а – Класично трижично поврзување (интегрирано напојување и заземјување со сензорите и актуаторите)



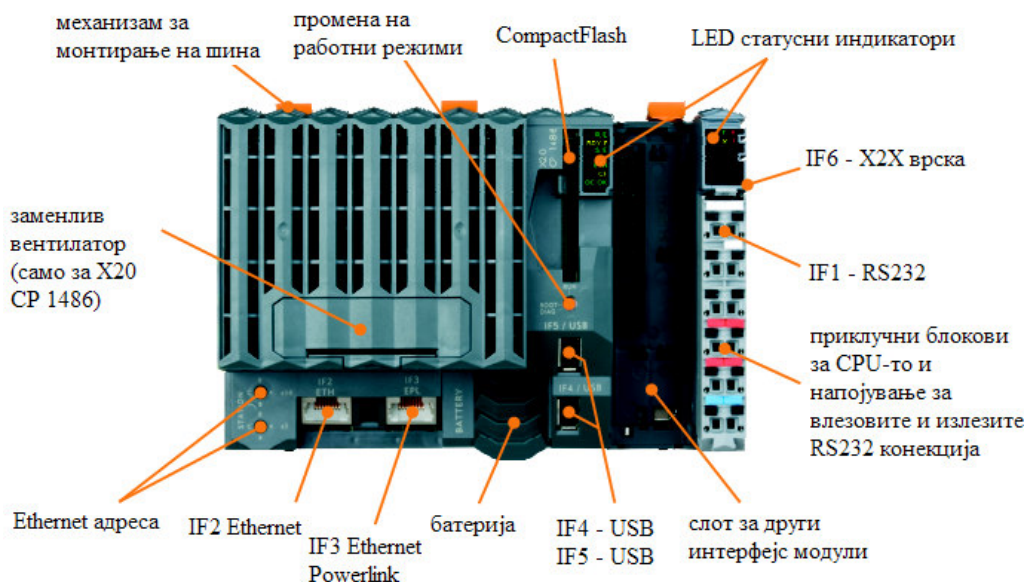
Слика 2.5б – Класично двојично поврзување (не се потребни дополнителни приклучоци)



Слика 2.5в – Класично едножично поврзување


## 2.2 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 2.6 е прикажан контролерот X20 CP1485 со составните делови.



Слика 2.6 – 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



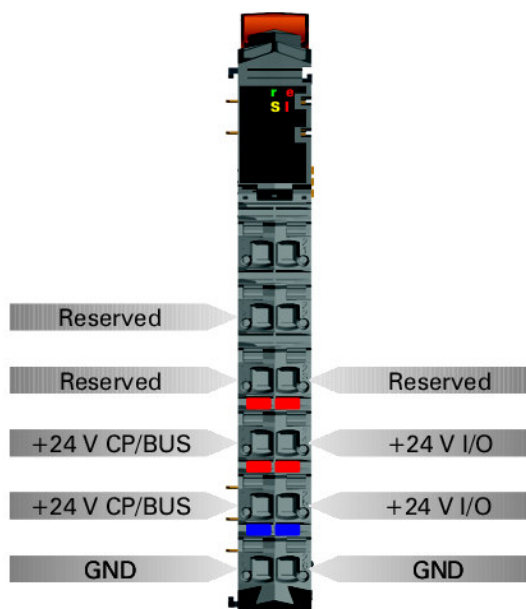
LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Неповрзано напојување на модулот.
		трепка еднаш	Ресетирање.
		трепка	Подготвителен режим на работа.
		вклучено	Работен режим (вклучено).
e	црвено	исклучено	Неповрзано напојување на модулот или се е во ред.
		трепка двапати	Означува еден од 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.

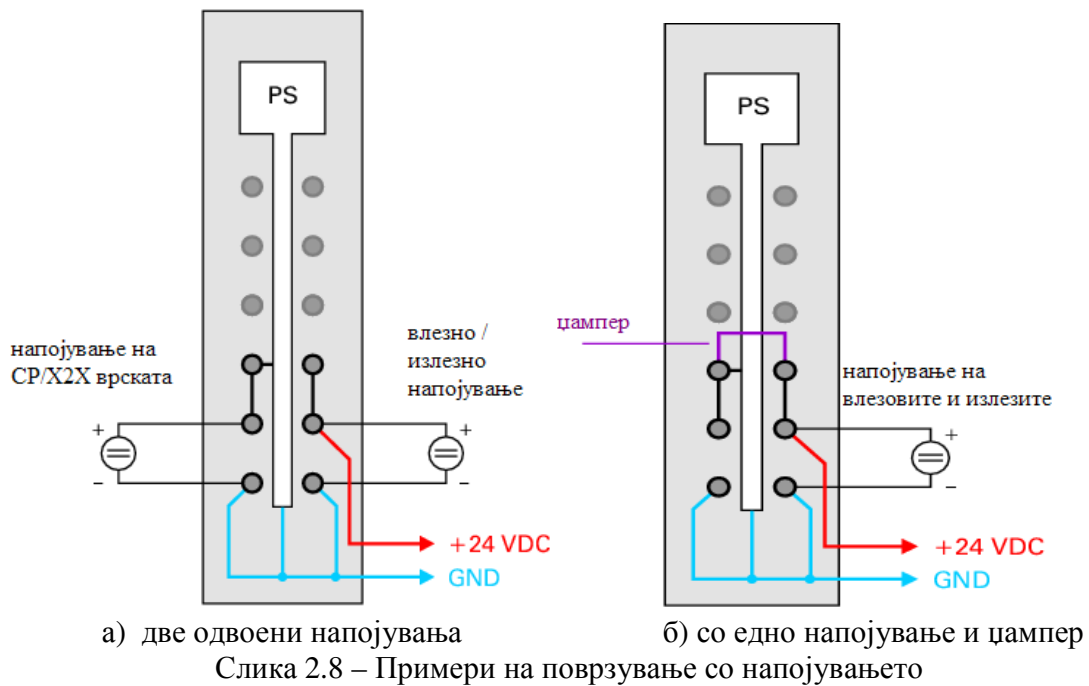
Со посебен прекинувач може да се променат три режими на работа на контролерот.

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).

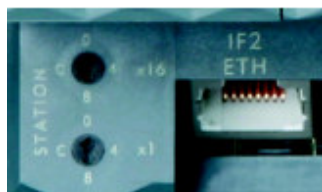
Напојувањето на процесорот доаѓа како интегрирано со X20 процесорите. Опремено е со напојување на процесорот, X2X врската и внатрешните влезови и излези. Напојувањето за процесорот и X2X врската е електрично изолирано.



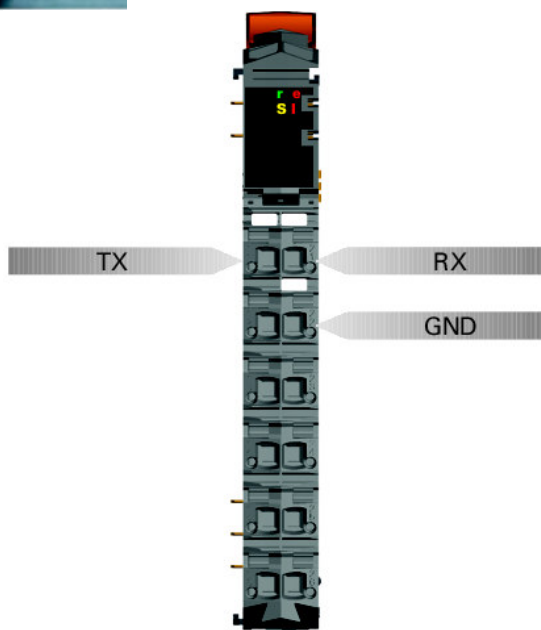
Слика 2.7 – Значење на пиновите на интегрираното напојување



Интерфејсот RS232 не е електрично изолиран. Тој се користи како online интерфејс за комуникација со уредот за програмирање (слика 2.9).



Ознаката IF2 го означува Ethernet интерфејсот за локални мрежи.



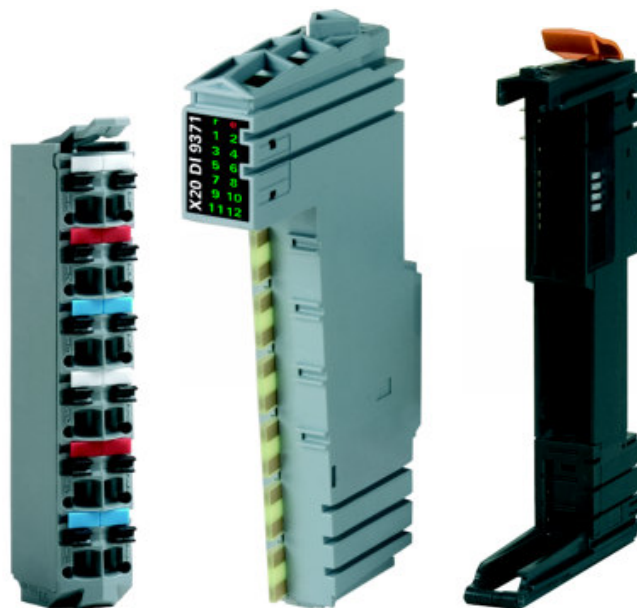
Слика 2.9 – Значење на пиновите на RS232 интерфејсот (IF1)

### 2.3. Дигитален влезен модул X20 DI9371

Дигиталниот влезен модул X20 DI9371 (слика 2.10) 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 2.5). 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 2.10).

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 2.10 се објаснети значењата на статусните индикатори на дисплејот.



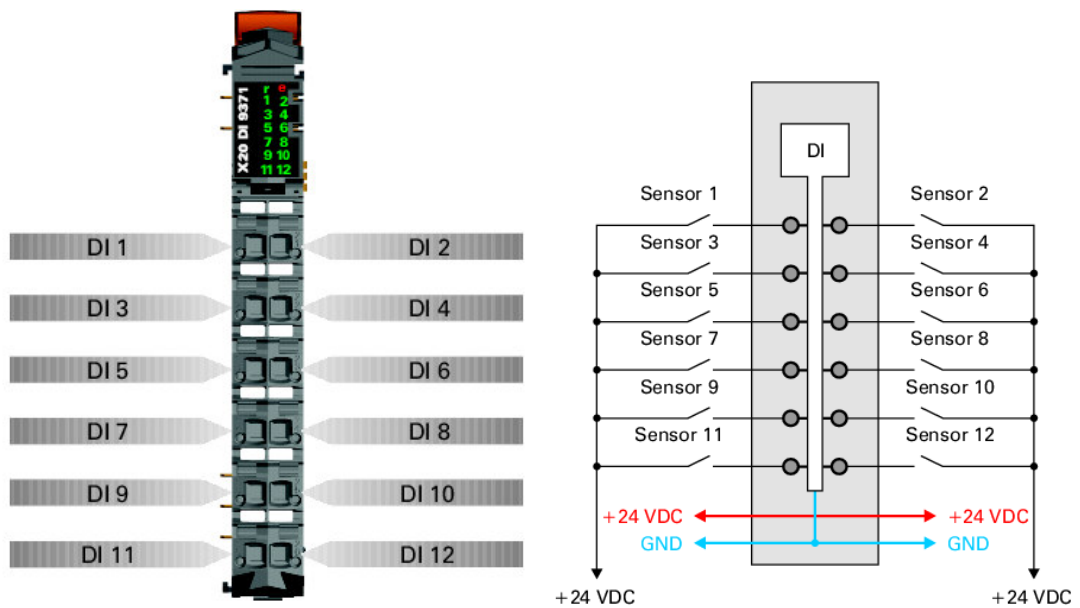
Слика 2.10 – Дигитален влезен модул X20 DI9371 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

### Статусни индикатори на влезниот модул

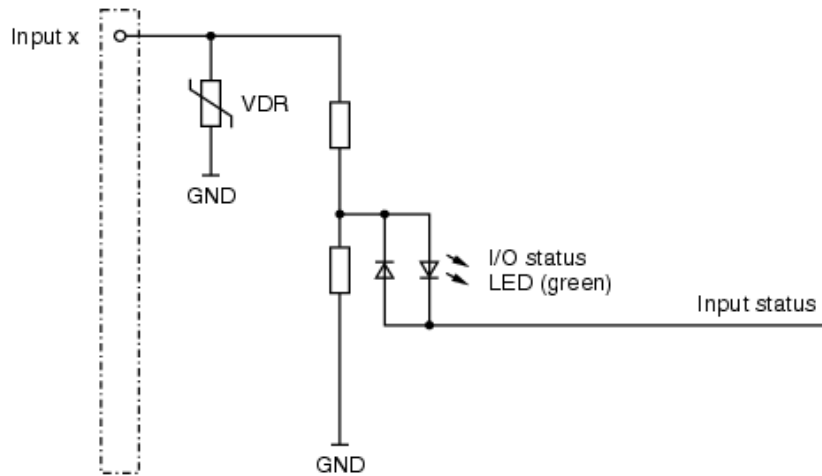


LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Напојувањето на модулот не е поврзано.
		трепка еднаш	Ресетирање.
		трепка	Подготвителен режим на работа.
		вклучено	Режим на работа.
e	црвено	исклучено	Напојувањето на модулот не е поврзано или се е во ред.
e + r	непрекинато црвено / зеленото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

На наредната слика 2.11 се прикажани значењата на пиновите, пример за едножичното поврзување, како и шема на влезното струјно коло.







Слика 2.11 – 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 влезовите и излезите е во врска со минималното време на запирање на циклусот, така што во секој циклус се случува ажурирање на влезовите и излезите. Тоа време за нефилтрирани сигнали е  $\geq 100 \mu\text{s}$ , а за филтрирани  $\geq 100 \mu\text{s}$ .

## 2.4. 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 регистрат 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 2.12).

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 2.12 се објаснети значењата на статусните индикатори на дисплејот.



Слика 2.12 – Аналоген влезен модул X20 AI4622 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 VM11 (десно)

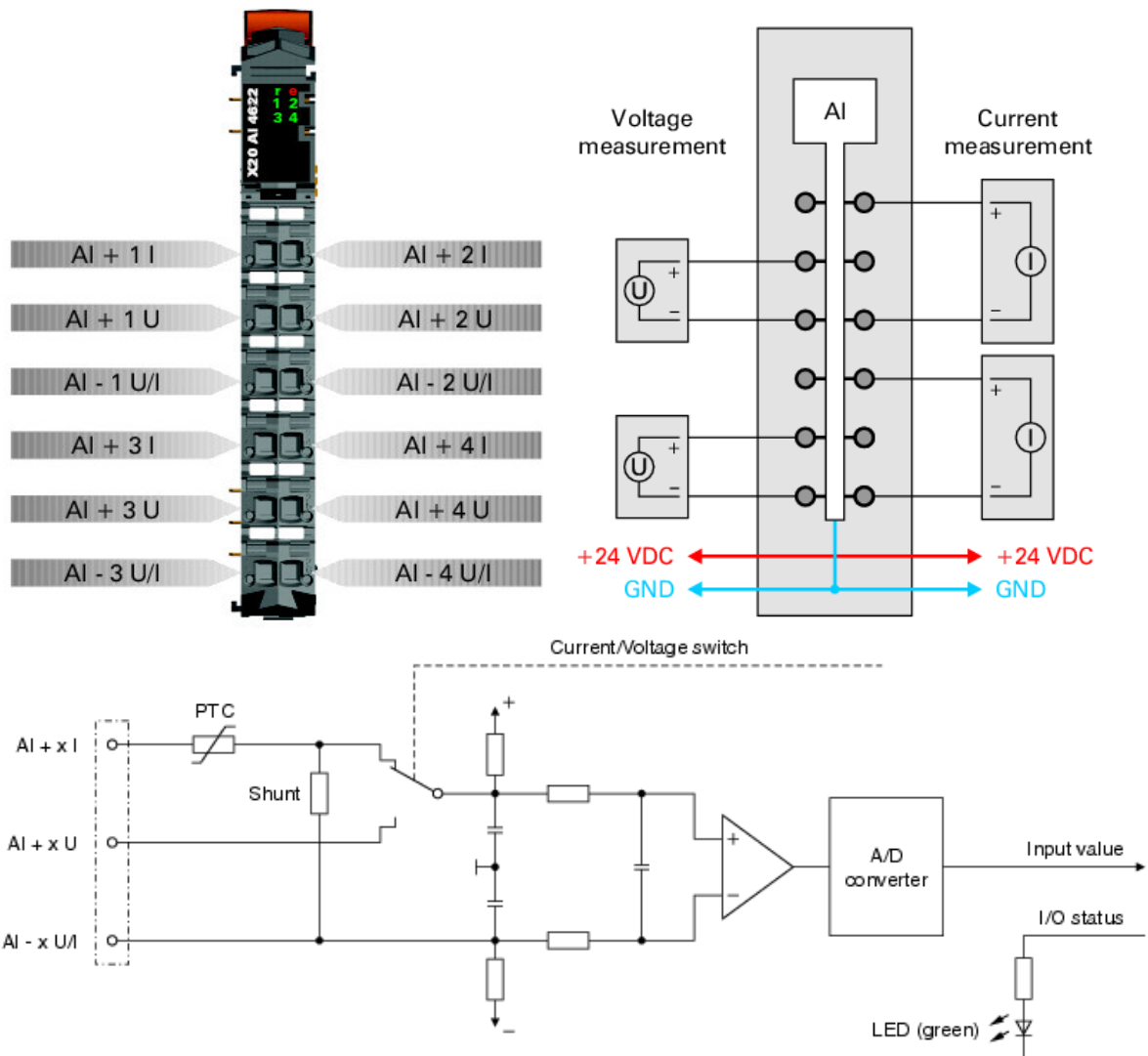
### Статусни индикатори на влезниот модул



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 2.13 – 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 500 $\mu$ s. 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 2.13). 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 (канали). 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.

## 2.5. 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 2.14).

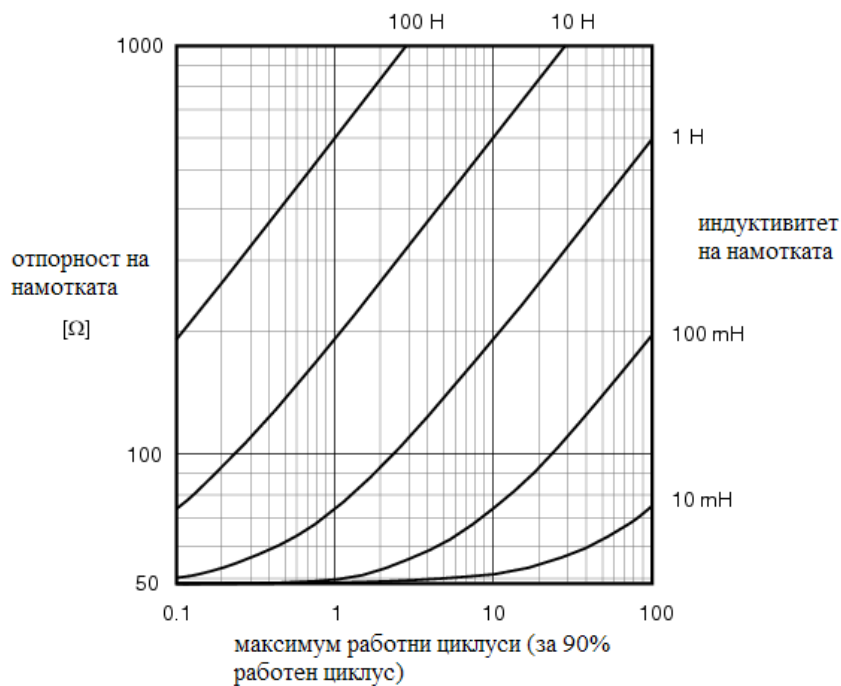


Слика 2.14 – Дигитален излезен модул X20 DO9322 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

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 2.15 се објаснети значењата на статусните индикатори на дисплејот.





Слика 2.15 – Вклучување и исклучување на индуктивни излези

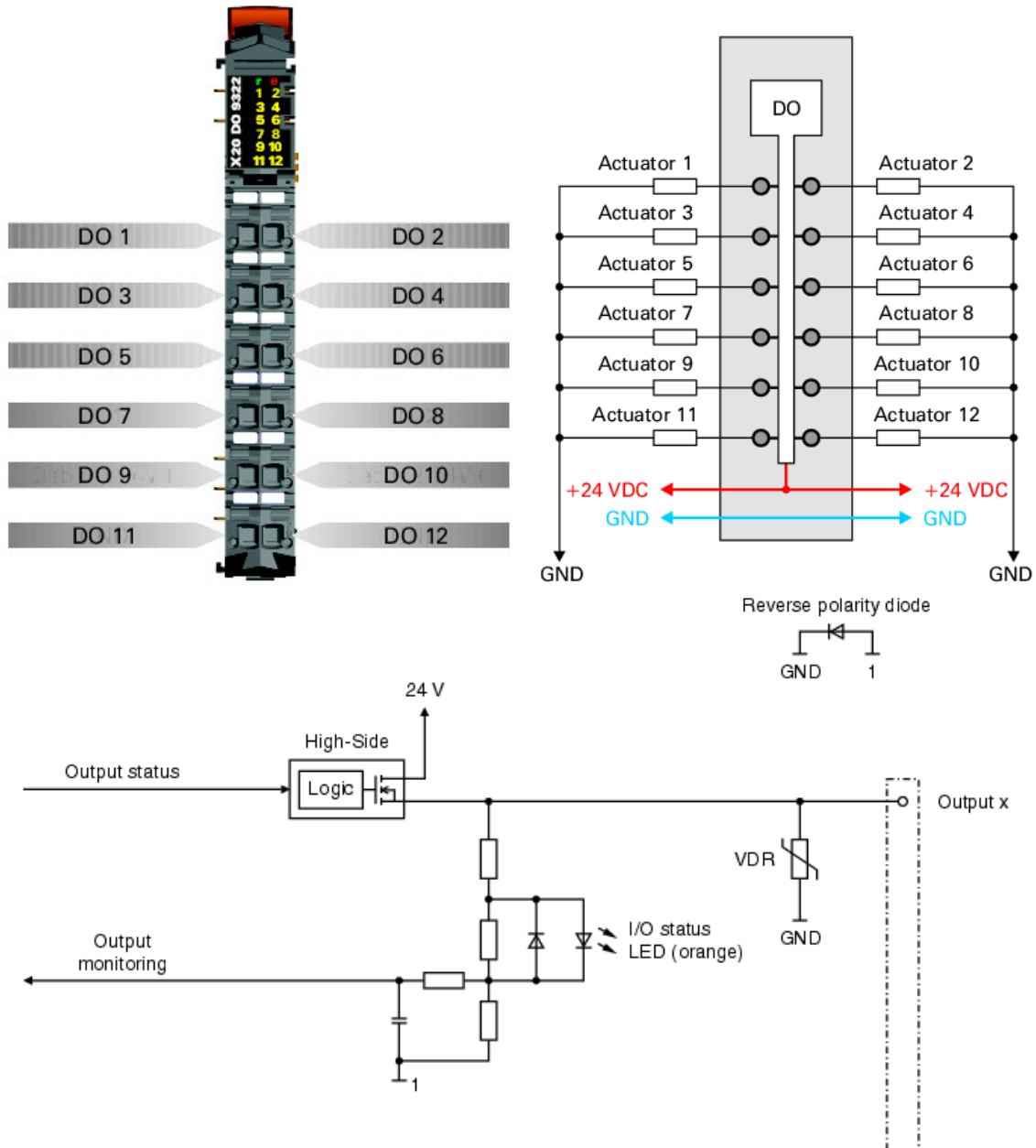
### Статусни индикатори на излезниот модул



LED индикатор	Боја на светлото	Статус	Опис
r	зелена	исклучено	Напојувањето на модулот не е поврзано.
		трепка еднаш	Ресетирање
		трепка	Подготвителен режим на работа
e	црвено	вклучено	Режим на работа
		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.

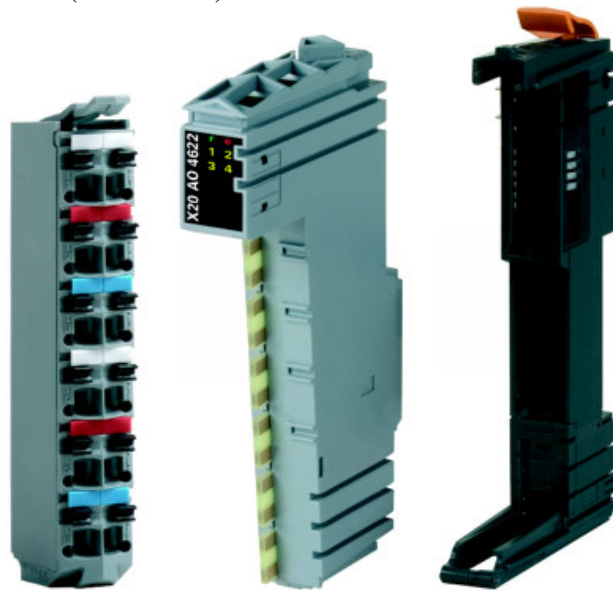


Слика 2.16 – Значење на пиновите, пример на поврзување со дигитални излети и шема на излезното коло

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.

## 2.6. 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 2.17).



Слика 2.17 – Аналоген излезен модул X20 AO4622 (средина) со приклучен блок X20 TB12 (лево) и магистрален модул X20 BM11 (десно)

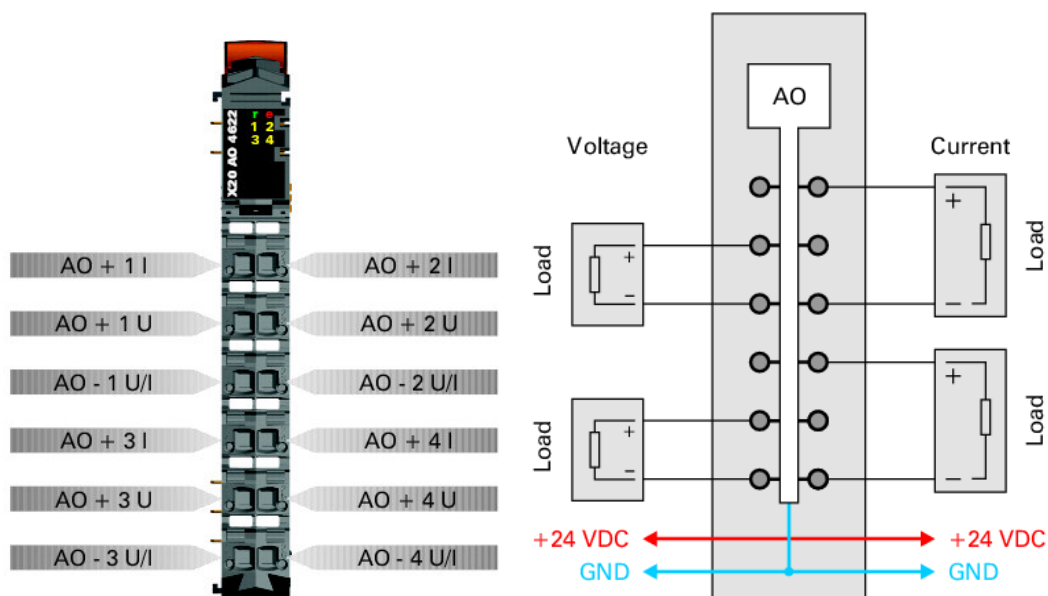
Излезниот напон е во границите од  $\pm 10$  V, а излезната струја во границите  $0 \div 20$  mA. Времето потребно за D/A конверзија е 300  $\mu$ s за сите излези. Излезите се заштитени од појава на краток спој со ограничување на струјата од  $\pm 40$  mA. Максимално дозволената отпорност на уредите 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  $1 \text{ LSB} = \frac{20}{2^{12}} = 9.766 \mu\text{A}$ .

Во наредната табела се објаснети статусните индикатори на дисплејот. На сликата 2.18 под табелата се прикажани значењата на пиновите и пример за поврзување. На слика 2.19 е дадена шема на излезното струјно коло.

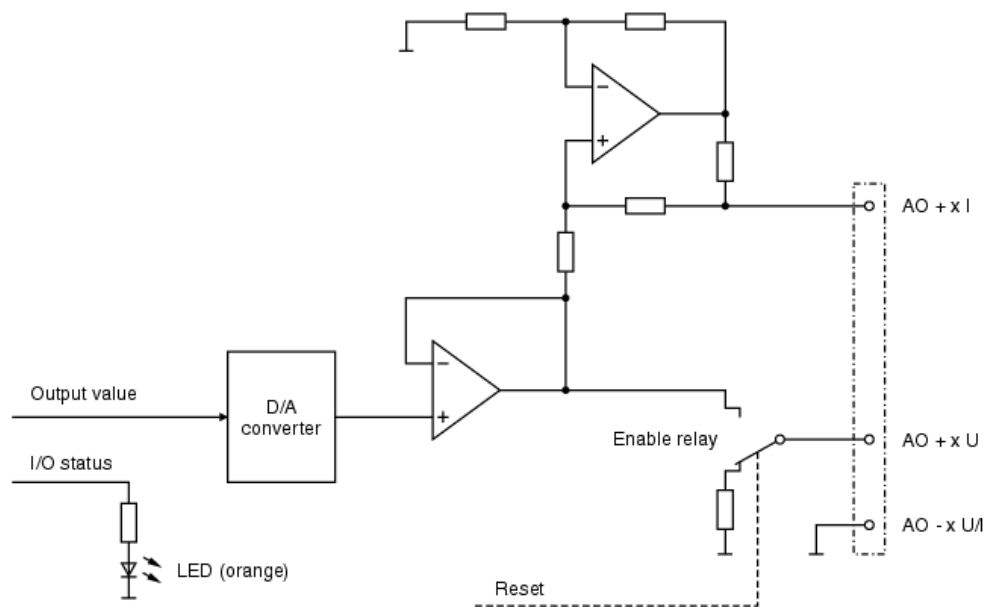
### Статусни индикатори на излезниот модул



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



Слика 2.18 – Значење на пиновите и пример на поврзување со аналогни излези



Слика 2.19 – 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 2.18). 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.

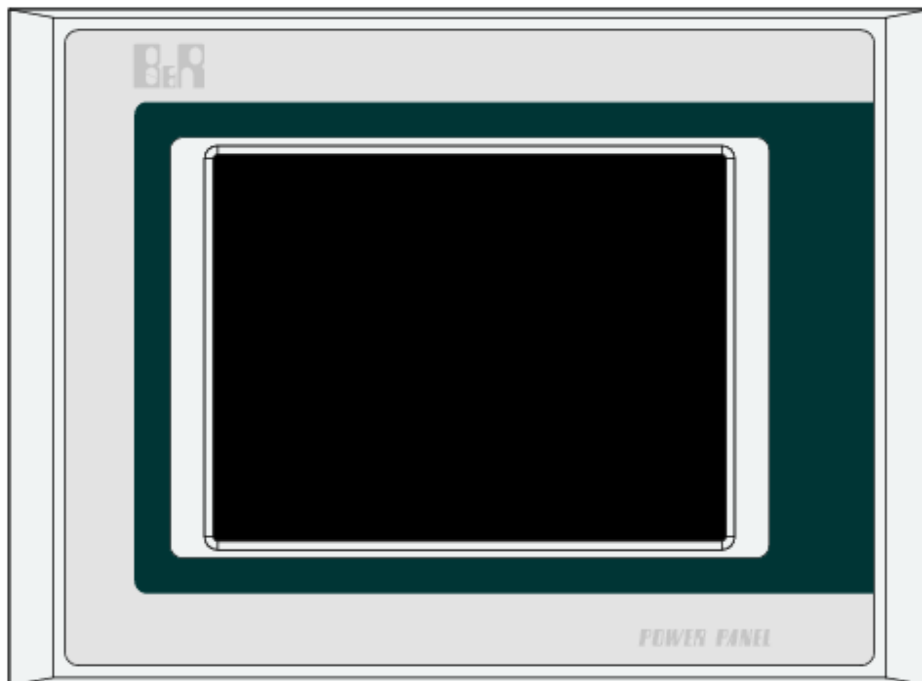
## 2.7. Операторски интерфејс модел 4PP320.0571.35 (Touch Screen)

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 2.20 и 2.21), исто така производство на V&R. Тој ќе биде искористен за управувањето на процесот, разработен подолу.

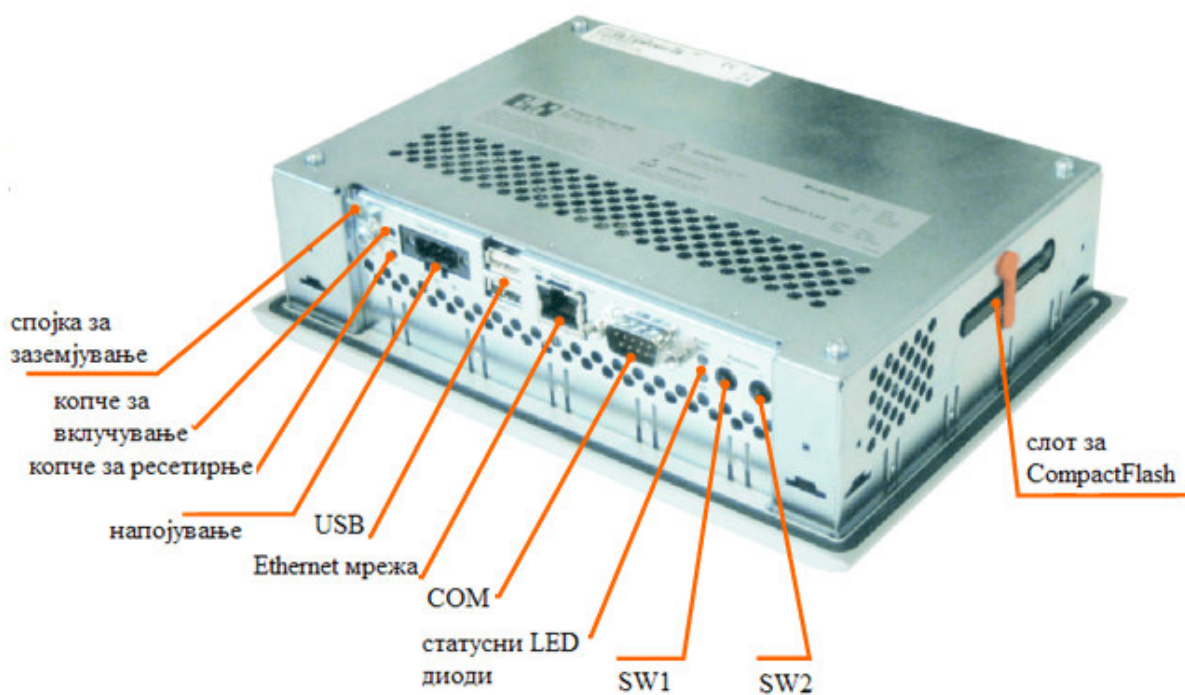
Во наредната табела ќе бидат прикажани некои технички спецификации на интерфејсот.



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 % $\pm$ 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 $\Omega$
<b>Механички карактеристики</b> Надворешни димензии Метално куќиште 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



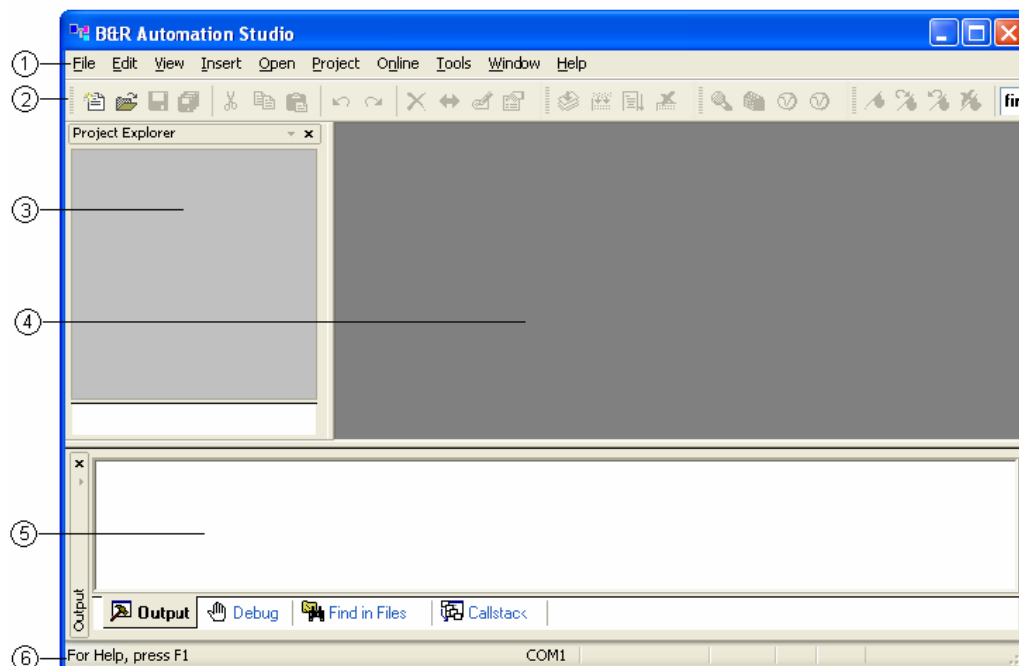
Слика 3.21 – Заден поглед на операторскиот интерфејс

### 3. Анализа на програмскиот пакет 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.

#### 3.1. 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 3.1.



Слика 3.1 – 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 B&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 менито **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.

## 3.2. 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

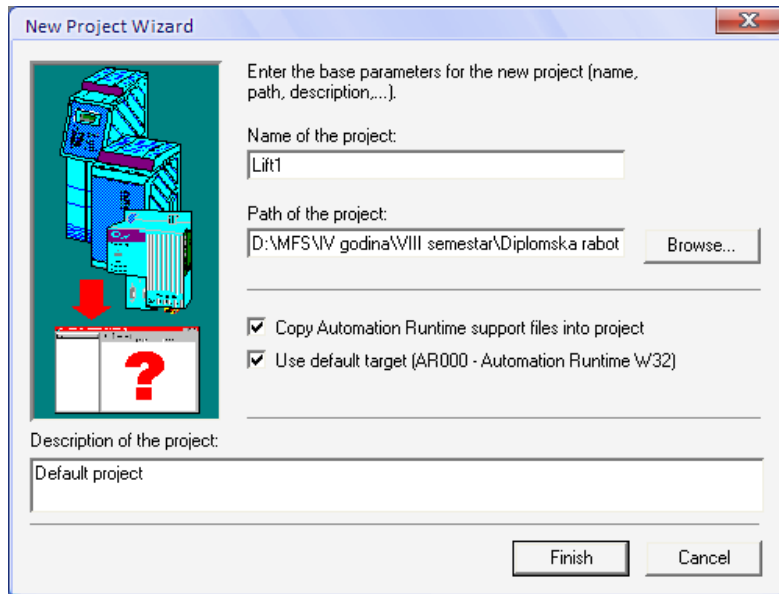
### 3.2.1. Постапка на креирање на проектот

Постапката на креирање нов проект започнува со менито **File / New Project**. Притоа се отвора прозорецот New Project Wizard како на слика 3.2.

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 3.2 – Креирање на нов проект

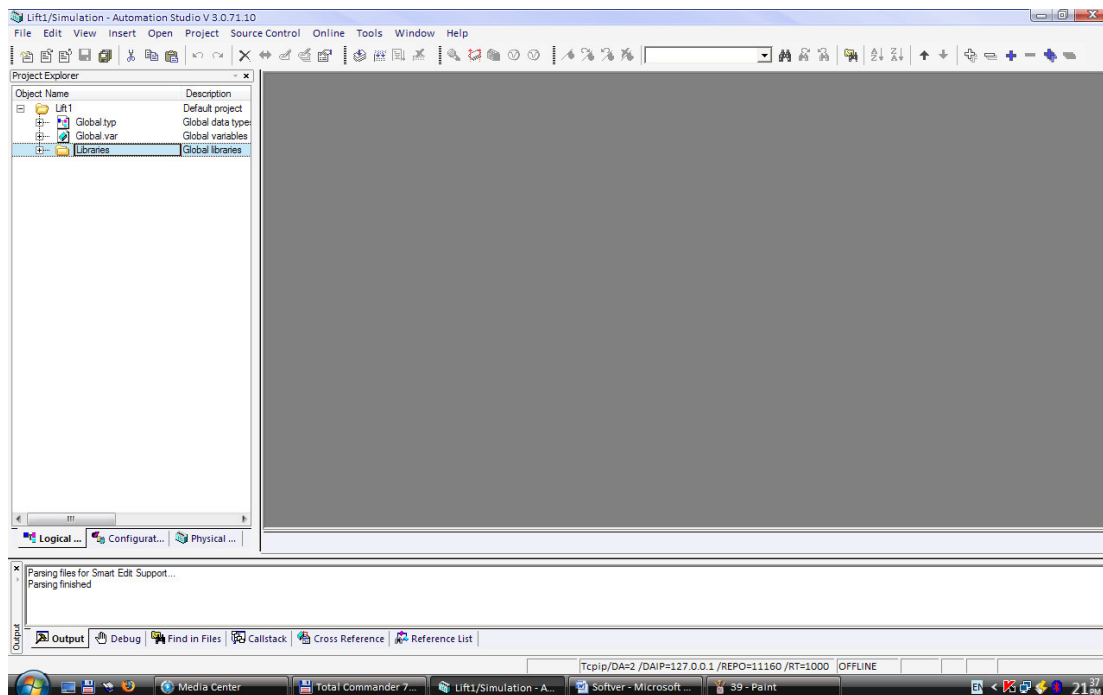
На крај се кликува на копчето **Finish** и проектот е креиран и прозорецот на програмата изгледа како на слика 3.3.

### 3.2.2. Креирање на програма

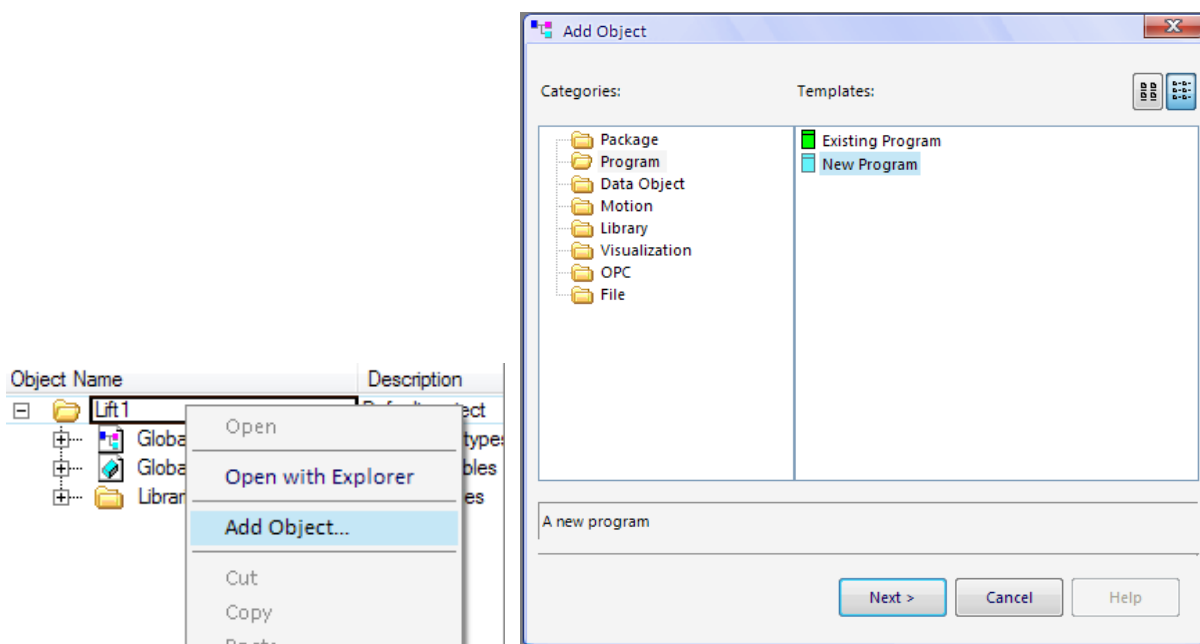
За да се внесе лидер дијаграм во проектот, потребно е да се превземат следните чекори:

- 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 3.4). 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**.



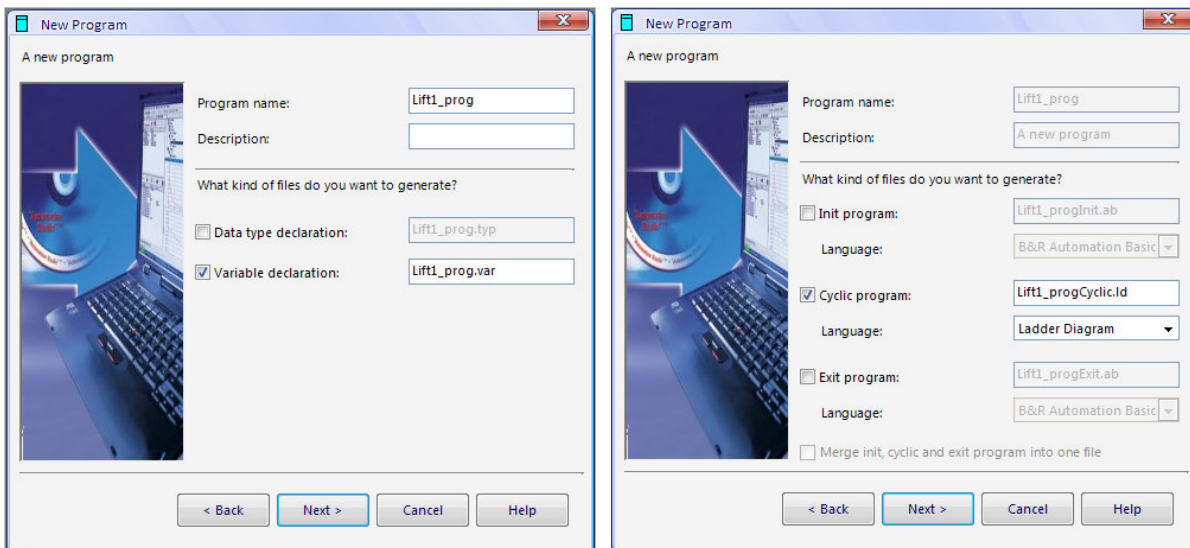
Слика 3.3 – Изглед на нов проект



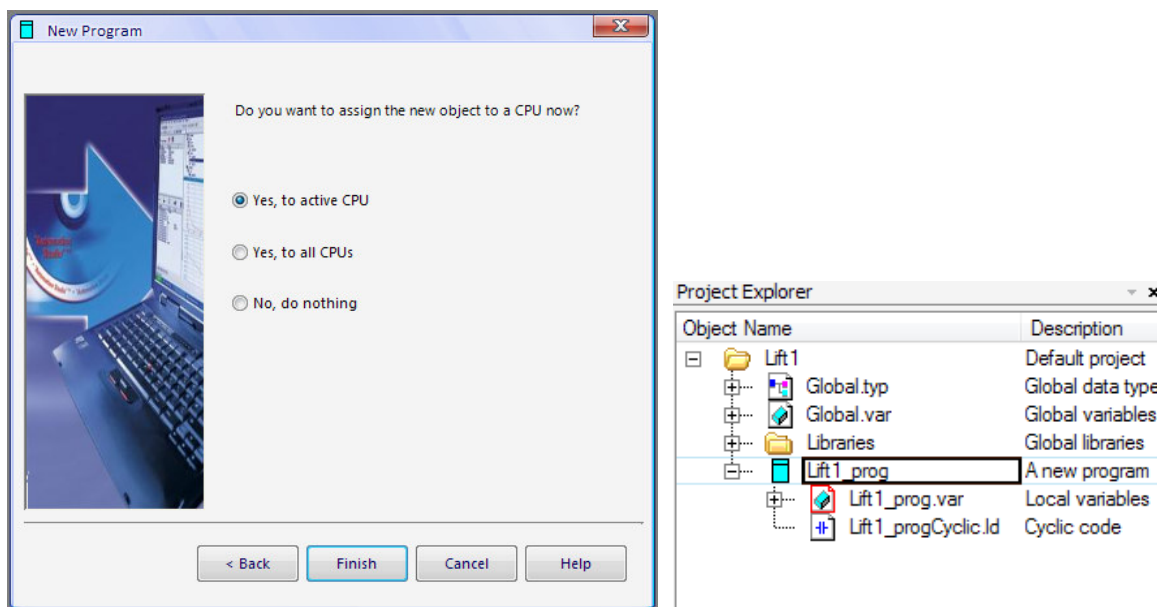
Слика 3.4 – Внесување на лидер дијаграм

Откако ќе се кликне на копчето **Next**, се појавува дијалог прозорец како на слика 3.5 – 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 3.5 – 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 3.5



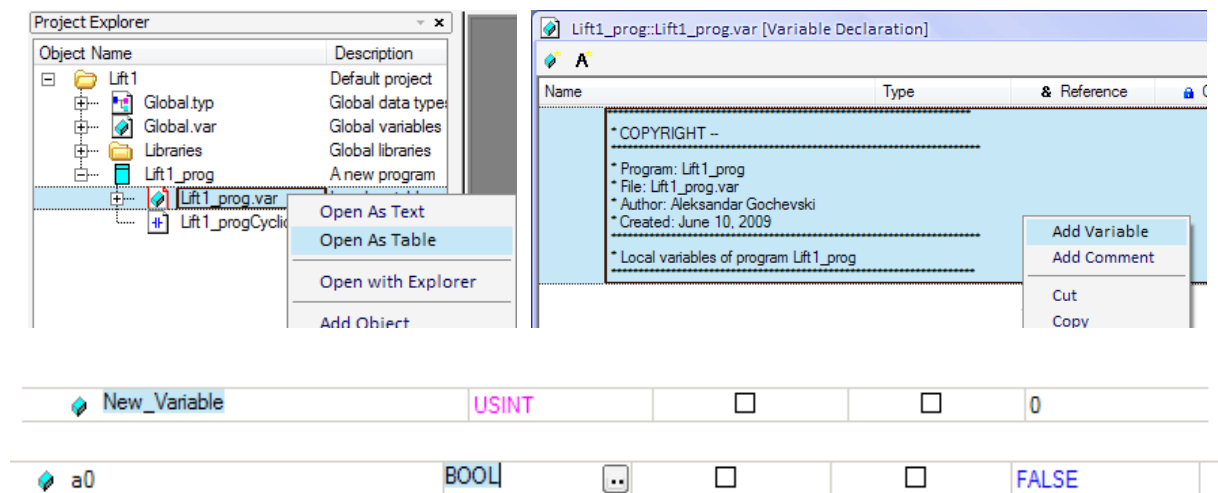
Слика 3.6

После притискање на копчето **Next** се појавува дијалог прозорец како на слика 3.6. Со доделување на објектот на централната процесорска единица, штиклирајќи ја опцијата



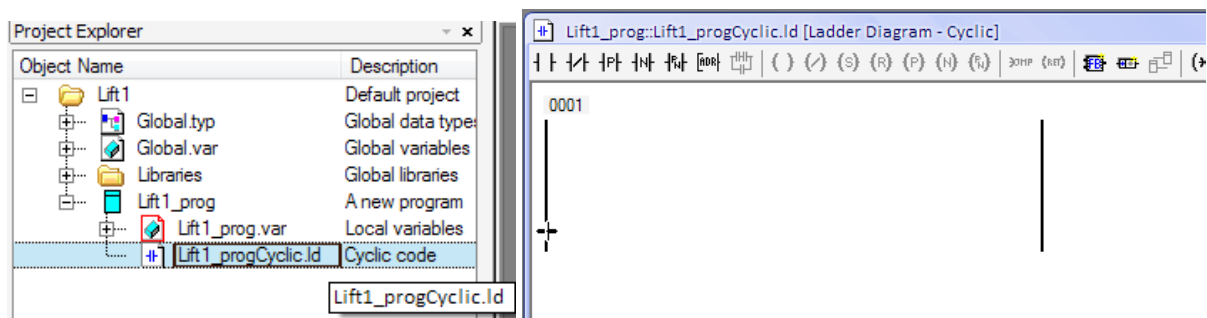
**Yes, to active CPU**, значи дека креираната програма е автоматски доделена на конфигурацијата на софтверот. Со копчето **Finish** завршува постапката на креирање нова програма и во Project Explorer проектот добива изглед како на слика 3.6 (десно).

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 3.7). 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 3.7

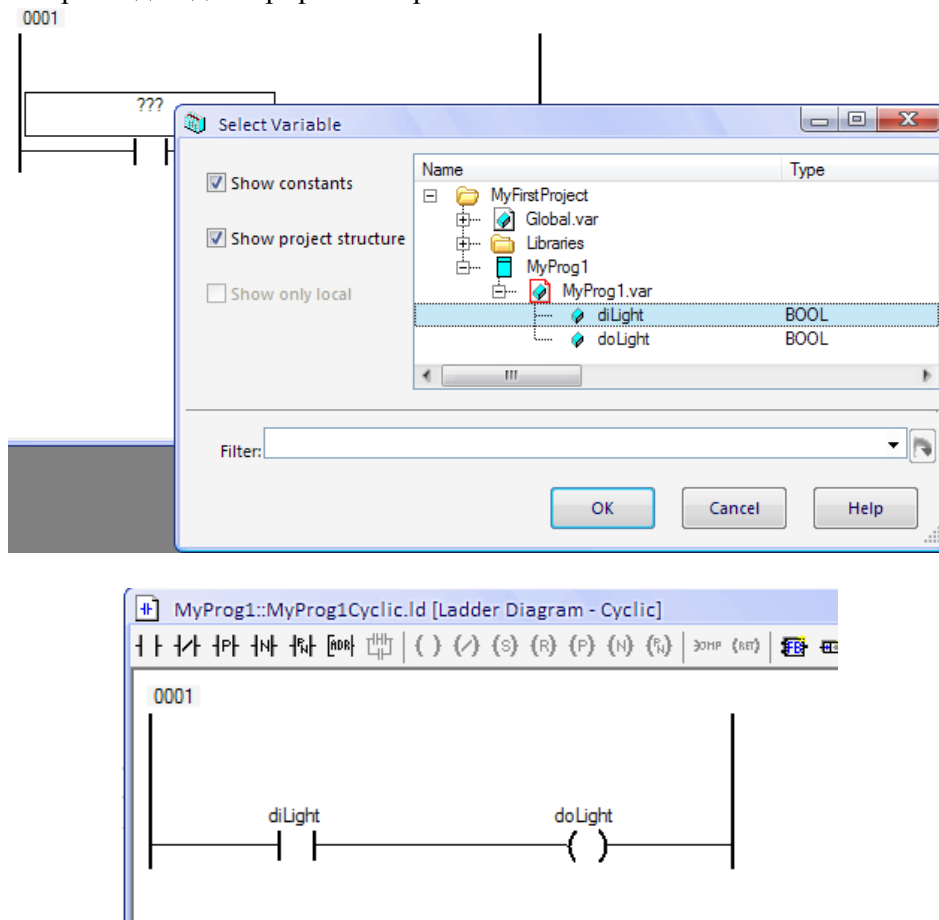
Откако ќе се дефинираат променливите, тие можат да се користат во ледер дијаграмот. За да се програмира ледер дијаграмот, потребно е да се отвори фајлот со екстензија .ld, што го содржи зборот Cyclic пред точката (**Lift1\_progCyclic.ld**) – слика 3.8. Во десната страна на главниот прозорец се појавува едитор за програмирање на ледер дијаграмот.



Слика 3.8

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 3.9, од каде што се одбира која од претходно декларираните променливи ќе се

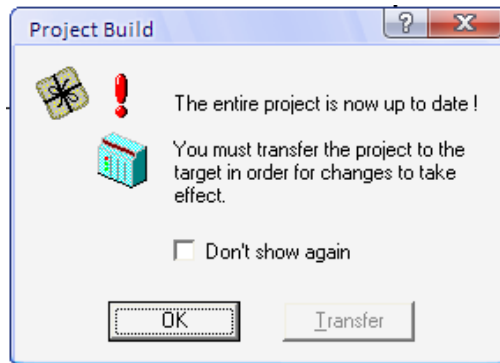
додели на контактот. На слика 3.9 е претставен пример на еден прост ледер дијаграм, користејќи ги претходно декларираните променливи.



Слика 3.9

### 3.2.3. 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 3.10. Тој исто така не информира дека проектот може да се пренесе до целниот систем (PLC – то или Automation Runtime за да се изврши симулација).



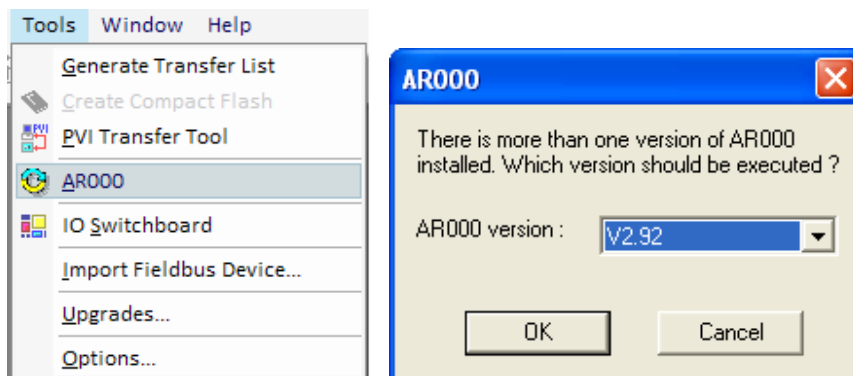
Слика 3.10 – Успешно преведена програма

### 3.2.4. Пренос на проектот кон целниот систем

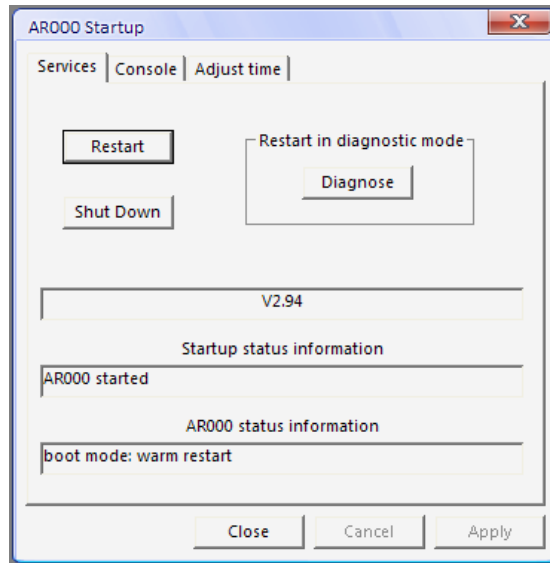
Во овој дел ќе се запознаеме како да се дефинира целен систем за еден проект. Во овој дел, наместо вистинското PLC, ќе го користиме софтверот Automation Runtime AR000. Со помош на овој софтвер, може да се изврши симулација на функционирањето на програмата пред таа да се префрли на PLC – то.

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 3.11 – лево). Доколку се инсталирани повеќе верзии на AR000, се појавува прозорецот на слика 3.11 – десно, во спротивно се појавува прозорецот на слика 3.12.

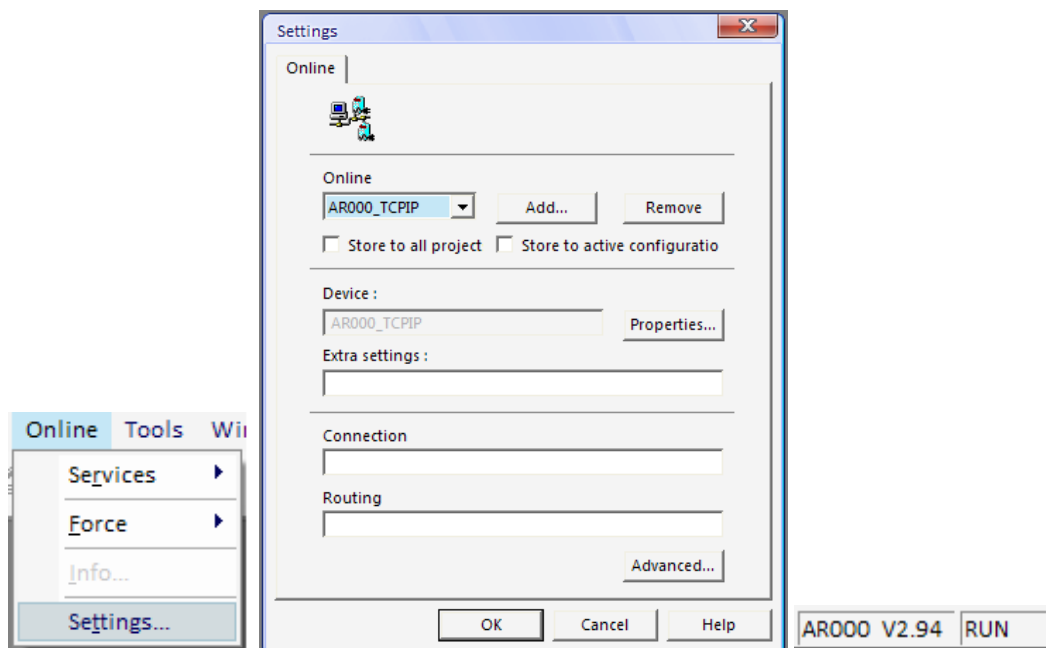


Слика 3.11



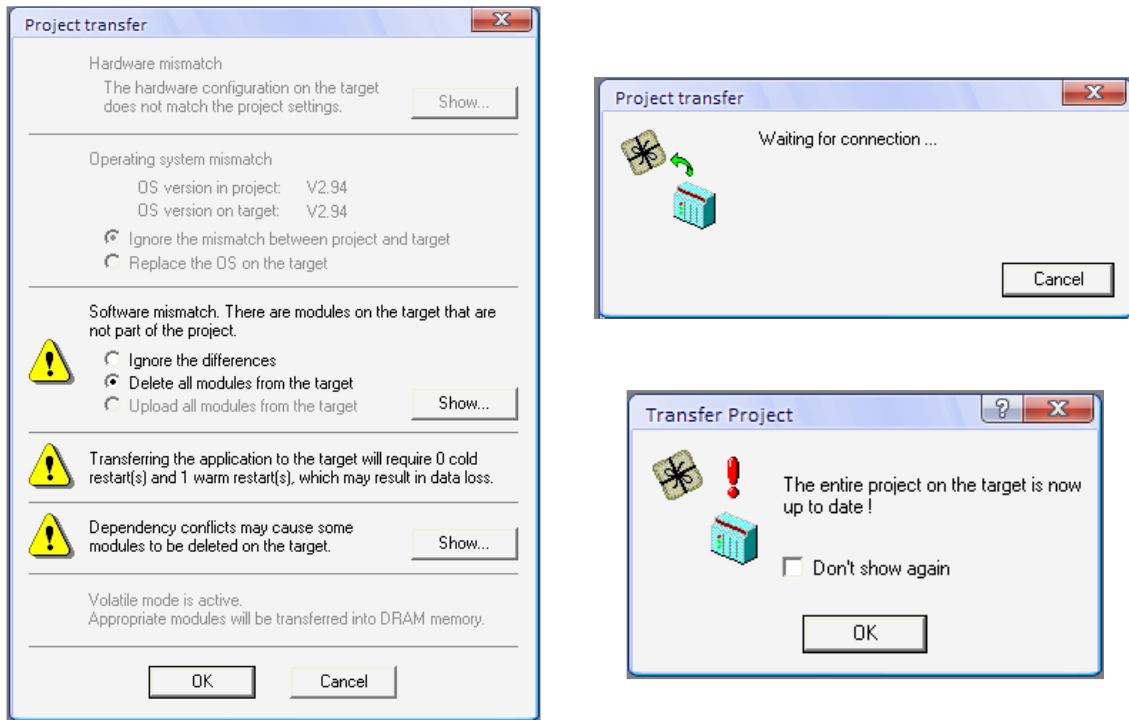
Слика 3.12

Откако ќе се појави прозорецот на слика 3.12, 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 3.13 – лево), по што се појавува прозорец како на слика 3.13 – средина. Подесувањето на конекцијата се прави така што се одбира **AR000\_TCPIP** од полето **Online**. Откако ќе се подигне AR000, на статусната линија (слика 3.13 – десно) наместо **OFFLINE** се појавува **RUN**, што означува дека врската со целниот систем е воспоставена.





Слика 3.13 – 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 3.14.

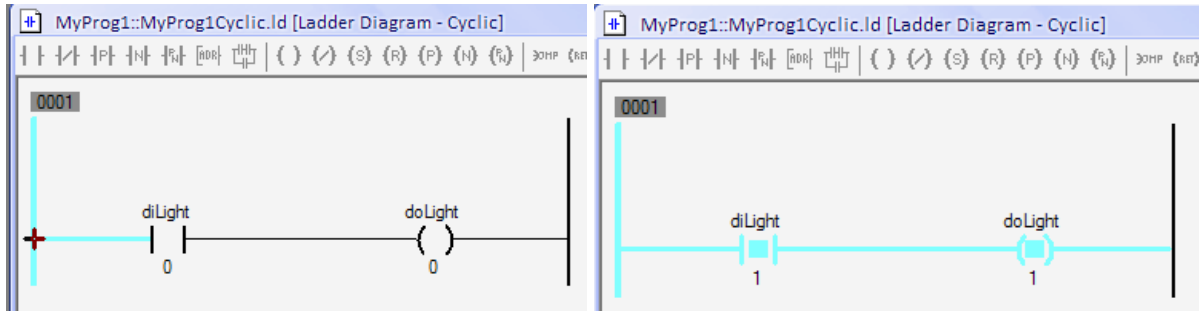


Слика 3.14 – Трансфер на проектот

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 ОК, започнува трансферот кон целниот систем, а дијалог прозорецот од слика 3.14 – 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.

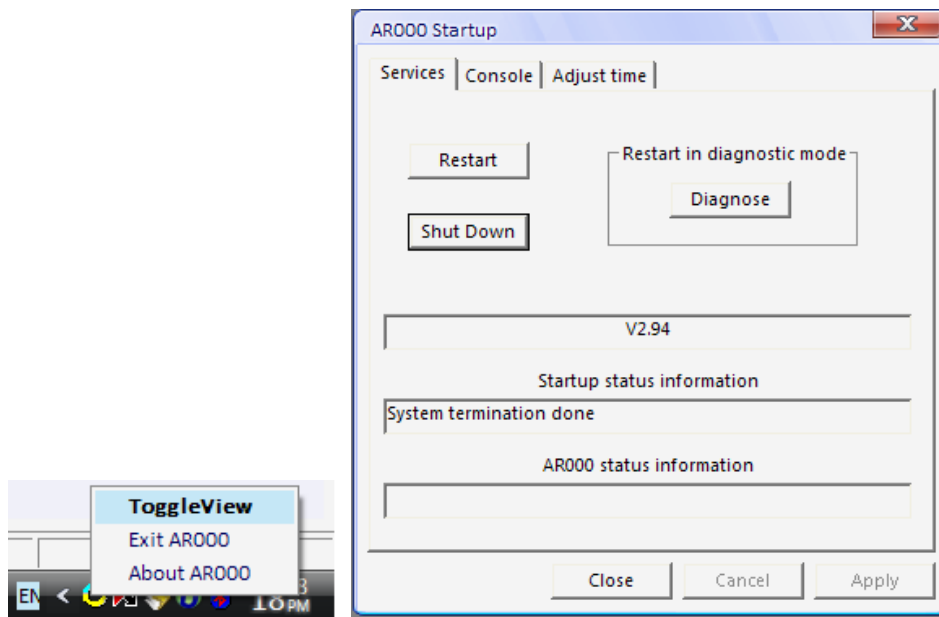
Тестирањето на лидер дијаграмот започнува со активирање на командата Monitor  од линијата со алатки, или со Ctrl+M од тастатура. Сега може да се провери дали со промена на влезните параметри, се добива посакуваниот излез. Текот на сигналите на лидер дијаграмот се вклучува со командата Powerflow  од линијата 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 3.15). 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 3.15 – Симулација на програмата со следење на текот на сигналите

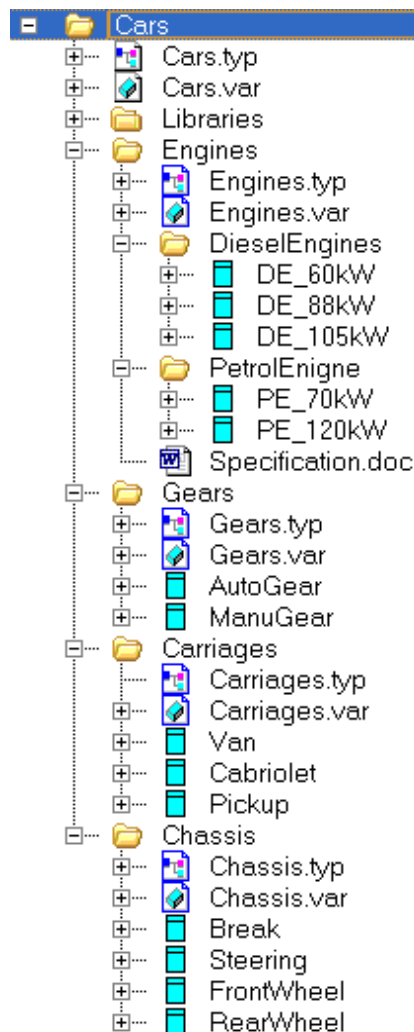
Откако ќе се изврши симулација на програмата, потребно е извршната програма на автоматскиот систем AR000 да се исклучи. Излезот од програмата мора да се направи правилно за да се запази конзистентност на податоците. Од линијата со активни процеси (taskbar), се кликува со десен клик на иконата на AR000 и се одбира ToggleView (слика 3.16). Потоа се појавува дијалог прозорец (слика 3.16) 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.



Слика 3.16 – Излез од Automation Runtime AR000

### 3.3. 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 3.17 е прикажан пример на организација на еден проект со помош на овој концепт.



Слика 3.17 – 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).

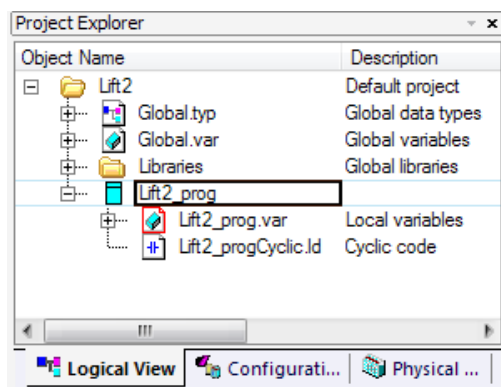
Под-дрвото Carriages (школки) исто така содржи дефиниции Carriages.typ и Carriages.var што се исти за сите типови на школки на возила, но истовремено со посебна функција (посебен софтвер) за секој тип на школка.

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.

### 3.3.1. 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.

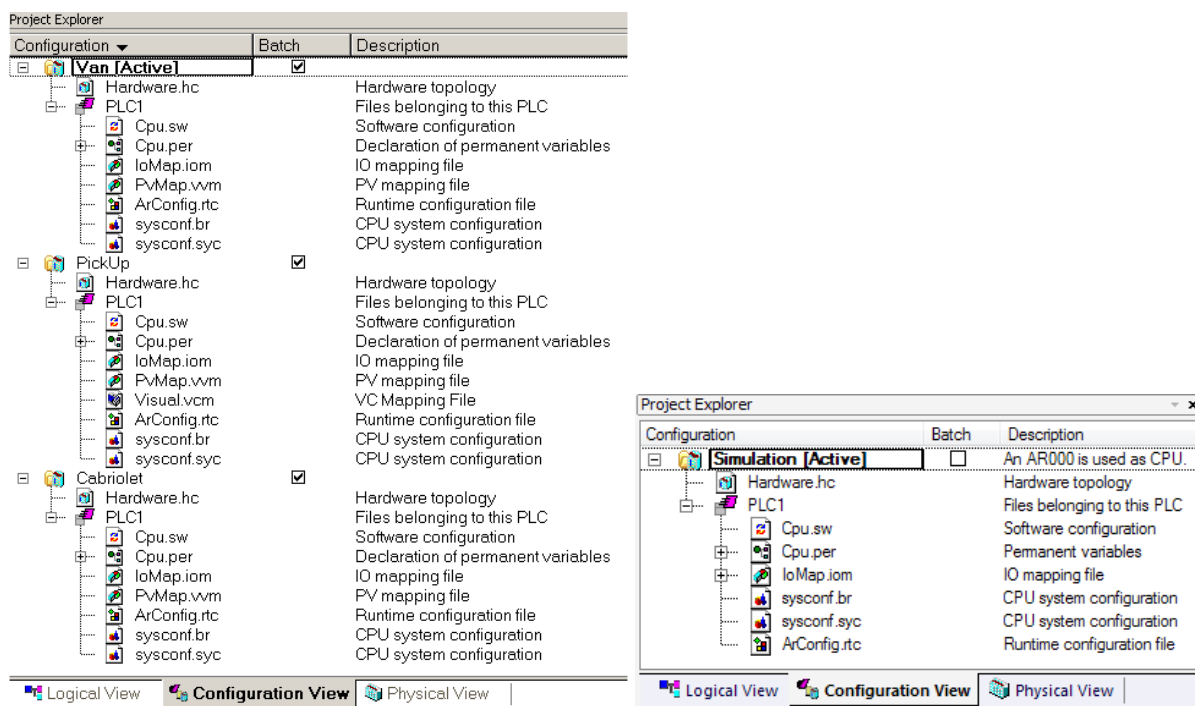


Слика 3.18 – Логички поглед на проектот Lift2

### 3.3.2. 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 3.19 е даден пример за конфигурациски 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 3.19 – Конфигурациски погледи

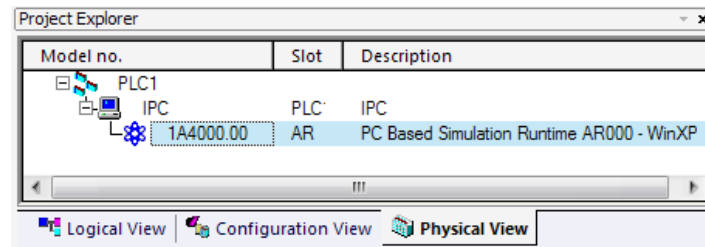
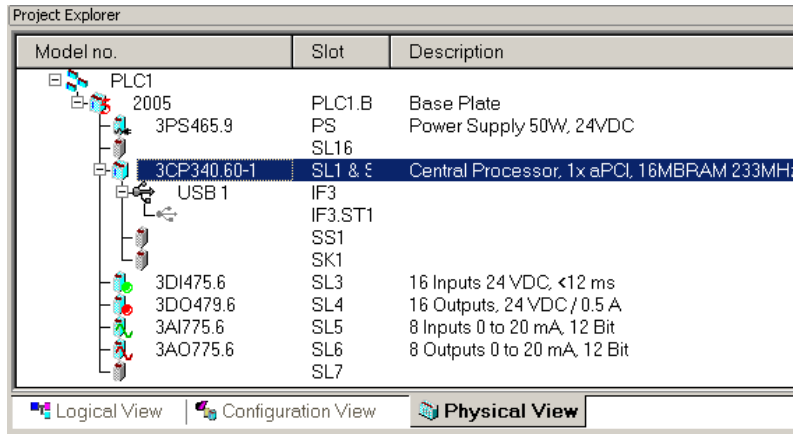
### 3.3.3. Физички поглед (Physical View)

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

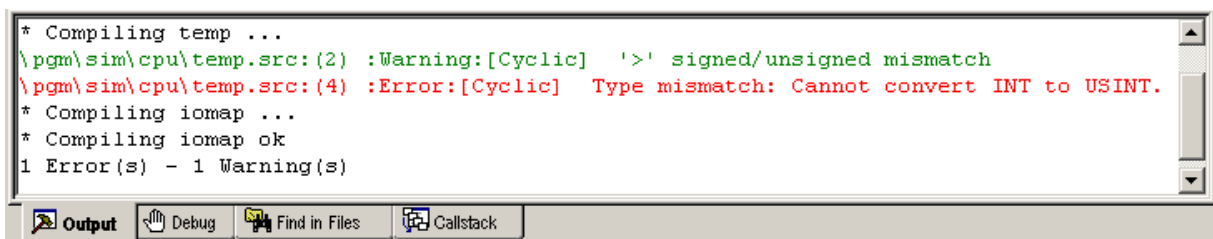
назначено хардвер, а единствениот физички изглед на активната конфигурација Simulation (слика 3.19) е Automation Runtime.



Слика 3.20 – Физички погледи

### 3.3.4. 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 3.21 – Излезен прозорец

Во излезниот прозорец се прикажуваат:

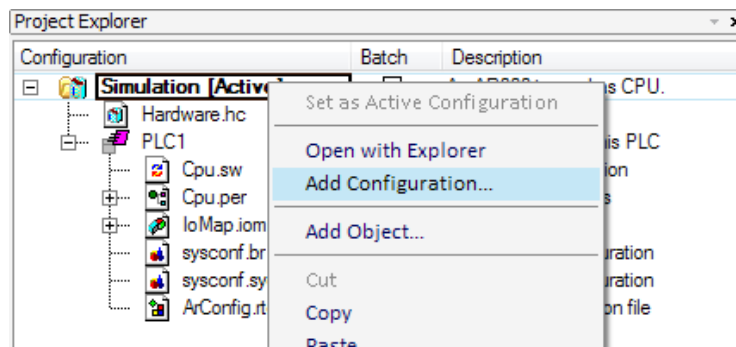
- Пораки на предупредувања и грешки за време на преведувањето на програмата. Двоен клик на пораката го носи курсерот на оној ред од програмата каде што се наоѓа грешката.
- Прогресот и статусот при пренесување на проектот до целниот систем.

- 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.

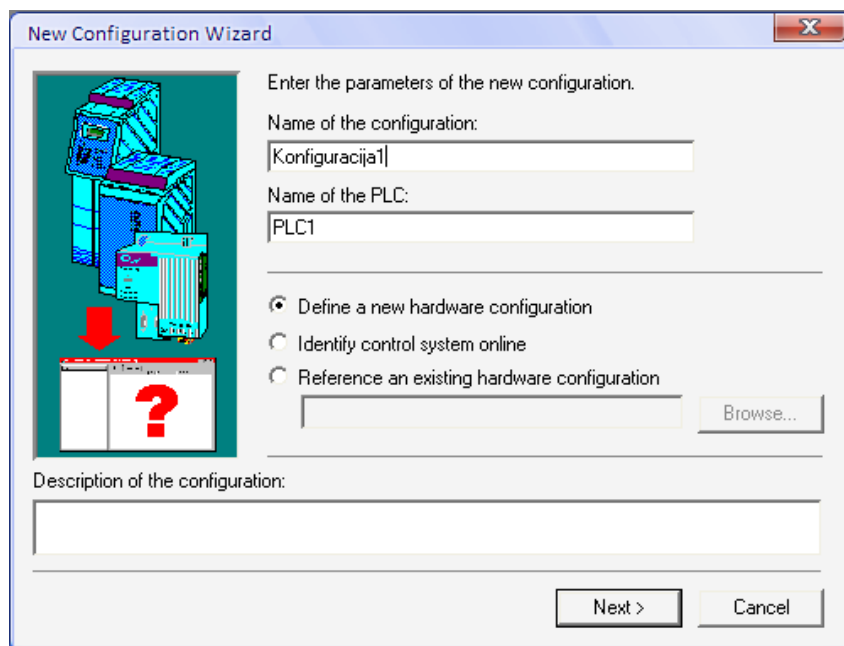
### 3.4. Подесувања на софтверот и хардверот во еден проект

#### 3.4.1. 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 3.19 – десно, т.е. содржи само конфигурација за симулација. Со десен клик во полето на Project Explorer (слика 3.22) се одбира опцијата **Add Configuration** и се појавува прозорец како на слика 3.23.

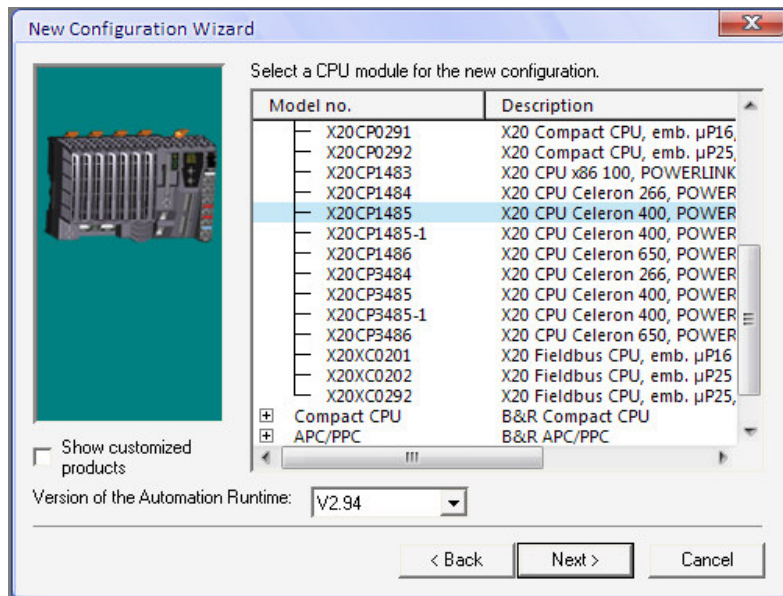


Слика 3.22



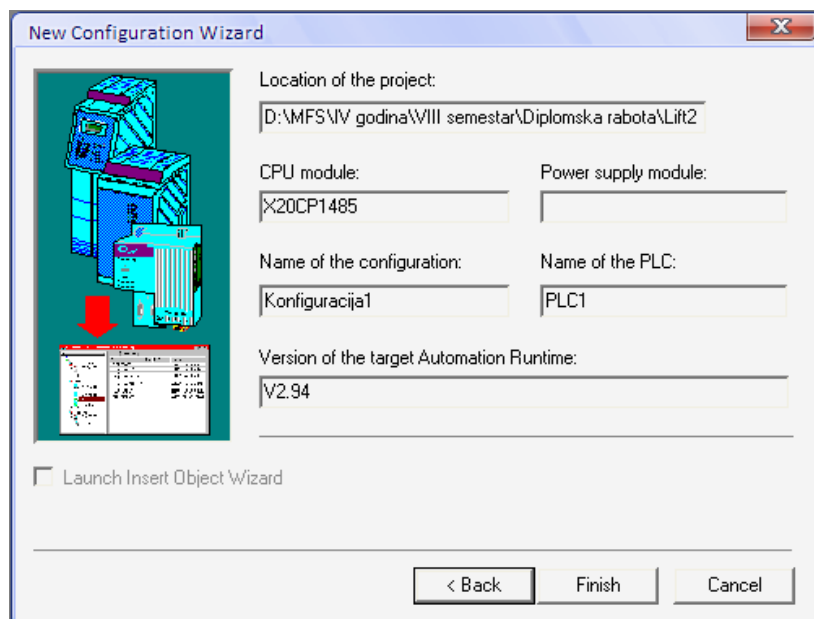
Слика 3.23

Во прозорецот како на слика 3.23 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 клик 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 кликнува **Next**.



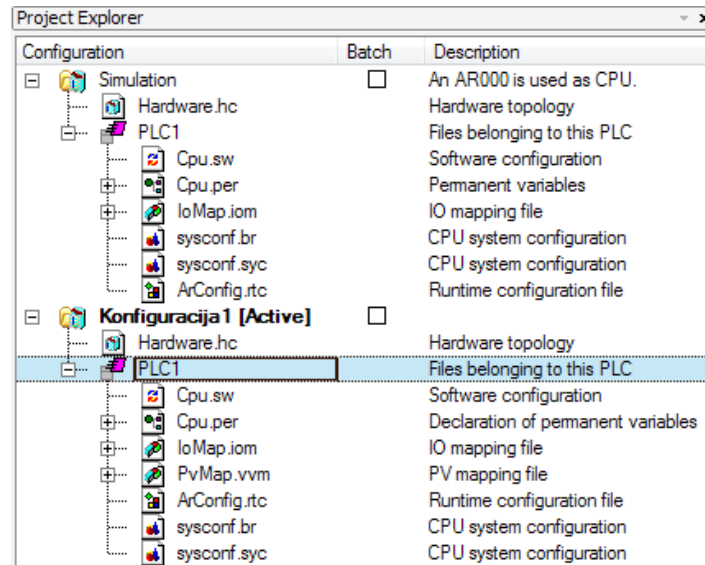
Слика 3.24

На следниот прозорец (слика 3.25) се прикажани сумирани опциите кои претходно сме ги избрале. Креирањето на конфигурацијата го завршуваме со клик на копчето **Finish**.



Слика 3.25

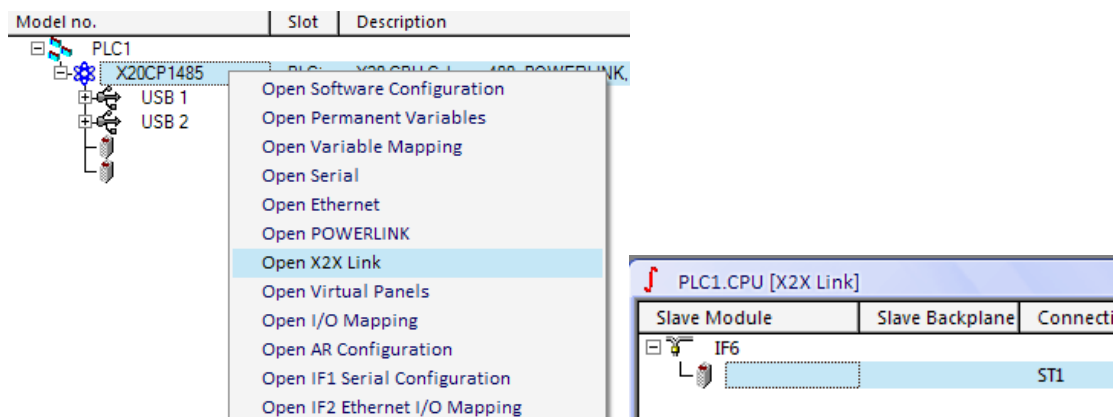
Новокреираната конфигурација „Konfiguracija1“ е успешно креирана. Конфигурацискиот поглед сега изгледа како на слика 3.26. Со двоен клик на името на конфигурацијата може да се менува активната конфигурација.



Слика 3.26

### 3.4.2. Додавање на потребниот хардвер

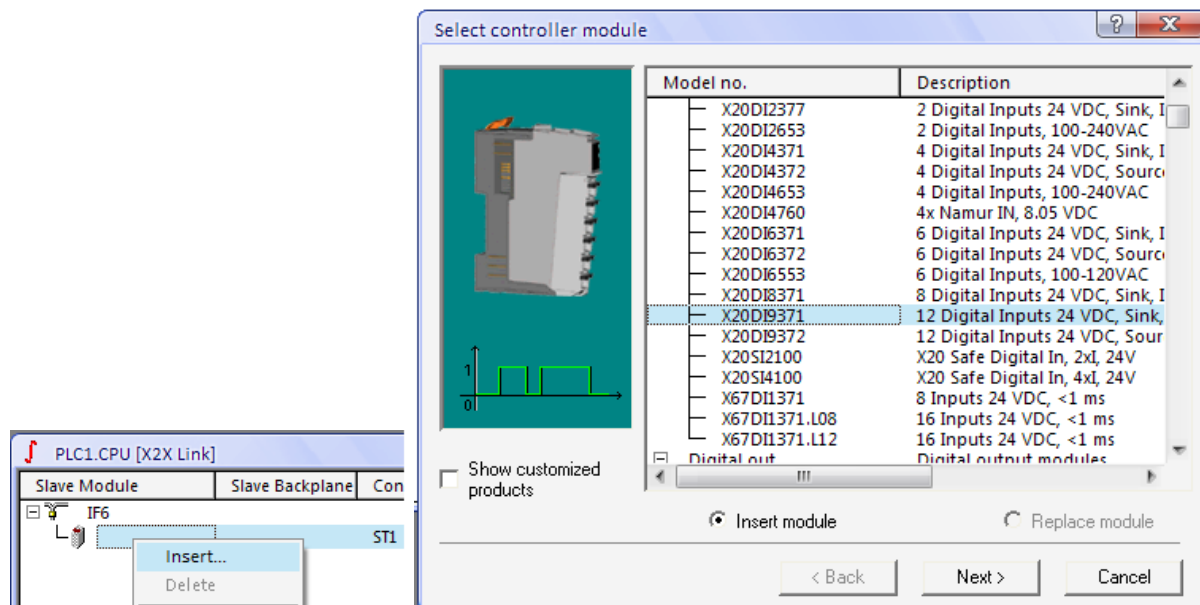
Хардверот што го содржи активната конфигурација е прикажан во физичкиот поглед (Physical View). Тука се додава хардверот потребен за реализација на задачата од автоматизација, во конкретниот случај, автоматизација на хидрауличниот лифт. Потребно е да се додадат следните модули: температурен модул X20 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 3.27 – лево.



Слика 3.27 – Отворање на X2X врската на X20 CP1485

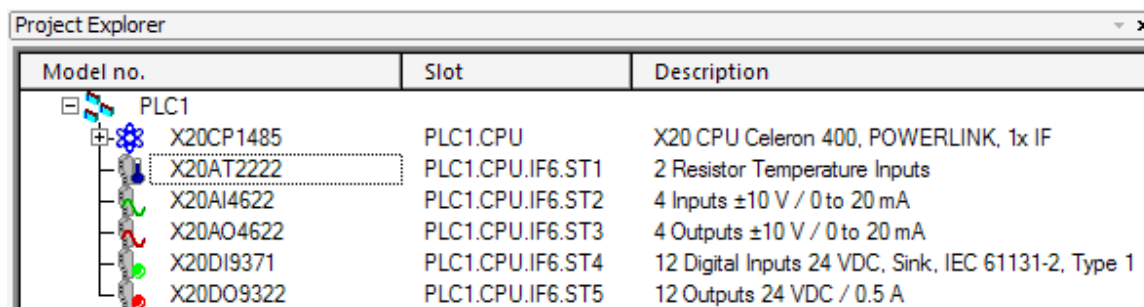
Тука се избира видот на врската помеѓу централната процесорска единица и модулите, а тоа е X2X врската. Откако ќе се кликне на **Open X2X Link**, се отвора прозорец како на слика 3.27 – десно и следи подесувањето на интерфејсниот приклучок IF6 (слика 2.6).

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 3.28). Од прозорецот на слика 3.28 – десно се избираат сите модули што треба да се додадат, повторувајќи ја постапката за секој модул одделно.



Слика 3.28 – Селекција на потребните модули

Откако заврши додавањето на хардверот, физичкиот поглед на проектот Lift2 изгледа како на слика 3.29.



Слика 3.29

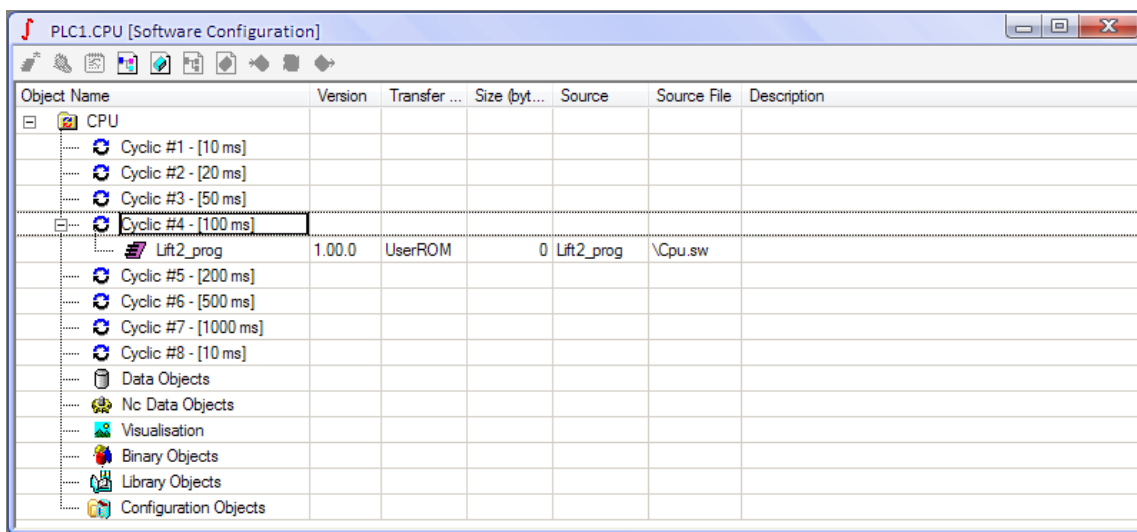
### 3.4.3. Софтверска конфигурација и доделување на софтверот

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 лифтоот ќе ја поставиме во циклусот број 4, со времетраење од 100 ms (слика 3.30).



Слика 3.30 – 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.

#### 3.4.4. 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 3.31). 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.

PLC1.CPU.IF6.ST1 [I/O Mapping]

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

Select

Variables

Use Data Type Filter

Data Type:

BOOL

Only Not Connected

Filter:

Name	Type
ProdluzvaKon1	BOOL
senzor1	BOOL
senzor1NegRab	BOOL
senzor2	BOOL
senzor3	BOOL
senzor3NegRab	BOOL
Spustaj	BOOL
Stop1	BOOL
Stop2	BOOL
Stop3	BOOL
Tajmer1	TON
Tajmer1_promenliva	BOOL
Tajmer2	TON
Tajmer2_promenliva	BOOL

OK Cancel Help

PLC1.CPU.IF6.ST2 [I/O Mapping]

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 3.31 – 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 3.31 (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.

## 4. 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.

### 4.1. 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 функциски групи.

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

Функциски блокови	√	√	√	√	√	√	√	√
-------------------	---	---	---	---	---	---	---	---

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

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

## 4.2. 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 својствени 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.

### 4.2.1. 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

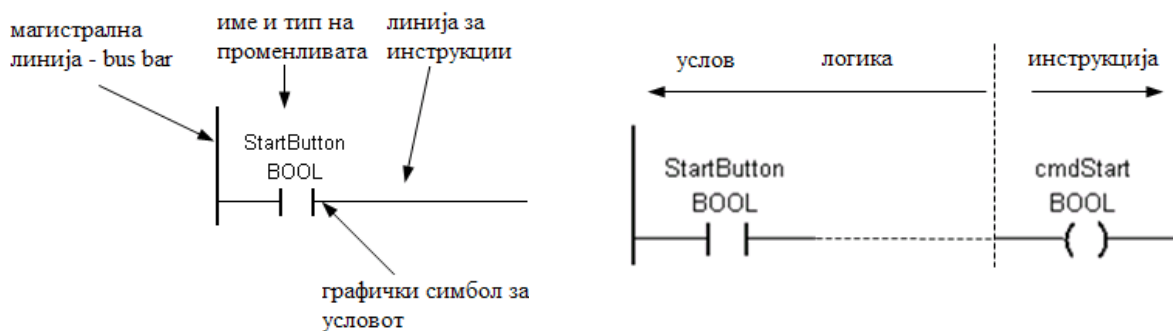
- интуитивен за употреба
- лесно се наоѓаат грешки
- се преведува (компајлира) е со IEC 61131-3 стандардот.

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 функционални блокови
- Контрола на текот на програмата (скокови)
- Алатки за дијагностика

#### 4.2.2. Основни елементи на ледер дијаграмот

Основните елементи на еден ледер дијаграм се прикажани на слика 4.1. 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.)



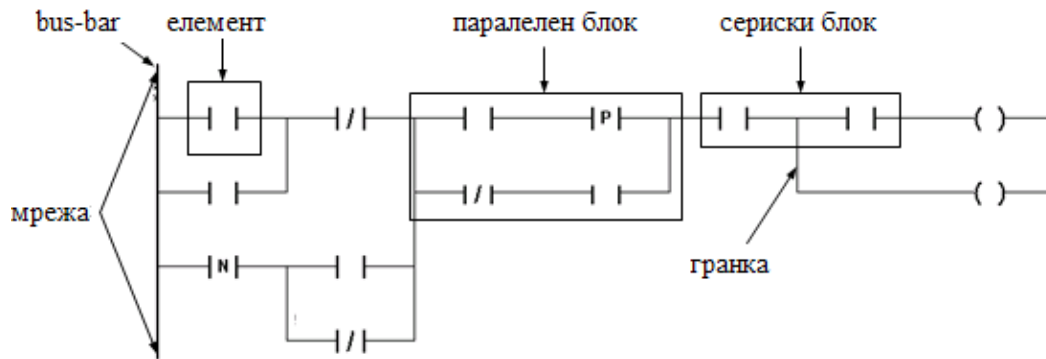
Слика 4.1 – Основни елементи на ледер дијаграмот

#### 4.2.3. 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 4.2 – Програмска мрежа на ледер дијаграмот

#### 4.2.4. Символи кај ледер дијаграмот

##### 4.2.4.1. 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 функциските 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.

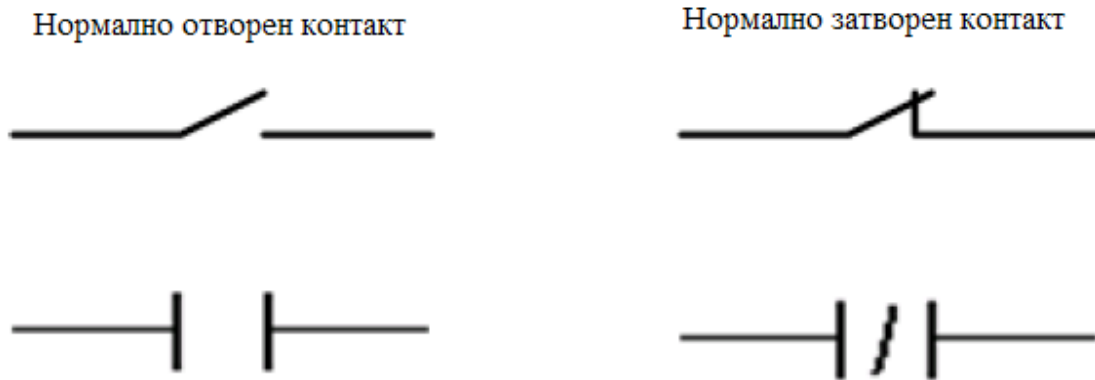
На еден контакт можат да се придружат само променливи од тип Boolean (BOOL).

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



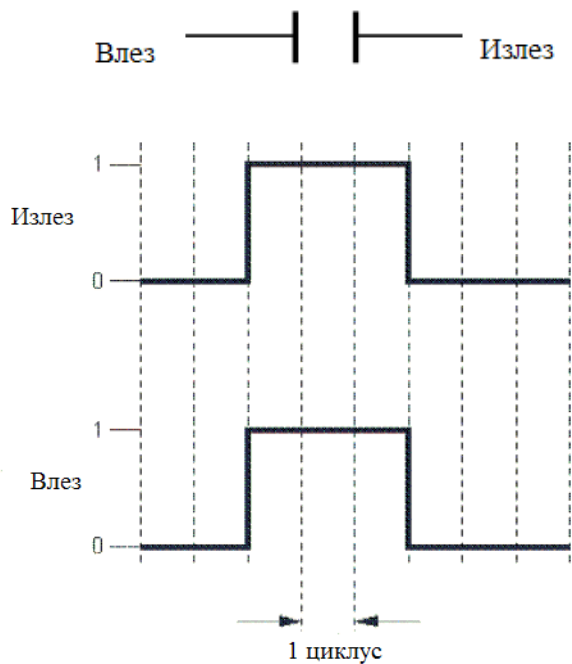
струја, ако е притиснат. Ако се отпушти тастерот, нормално отворениот контакт ќе престане да спроведува струја. Кај нормално затворениот контакт (на пример свонче) е обратно – тој ќе спроведува струја (свончето ќе свони), се додека не се притисне прекинувачот на свончето, со кој ќе се прекине струјното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

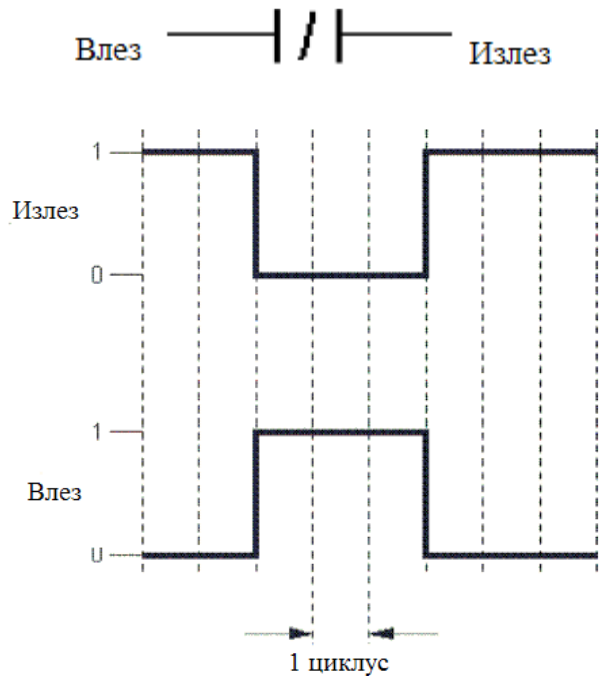
### 1. Нормално отворен контакт



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

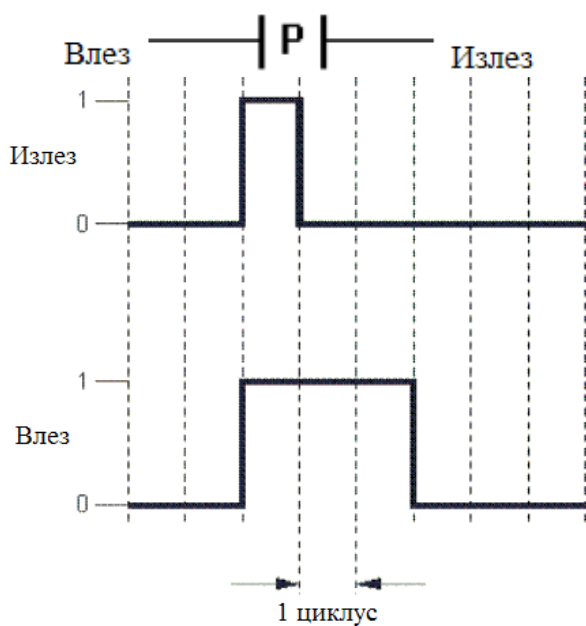


Овој симбол го менува статусот на променливата од тип Boolean.

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.

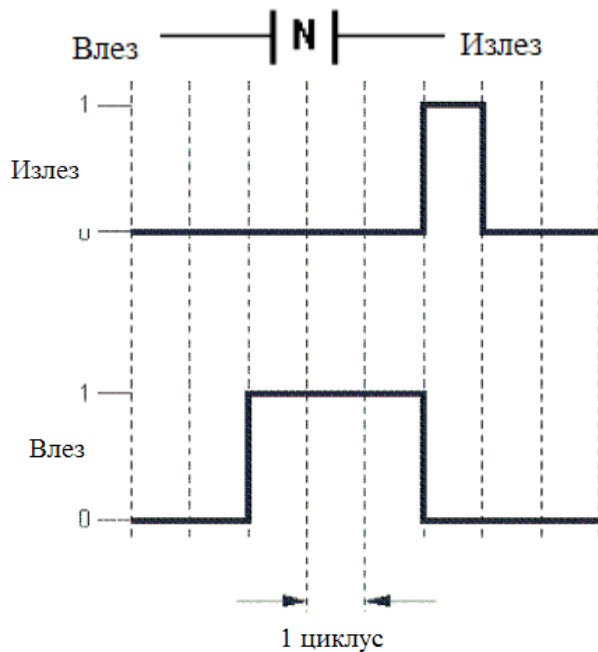
## 3. Позитивен раб



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.

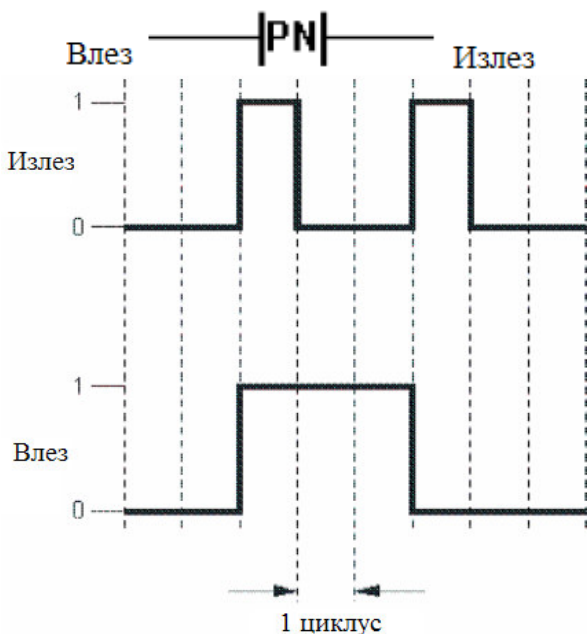
#### 4. Негативен раб



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).

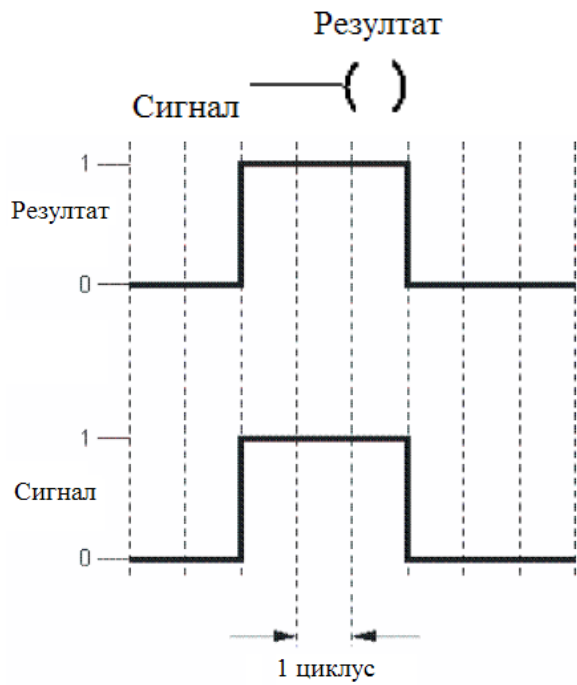
#### 4.2.4.2. Намотки

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.

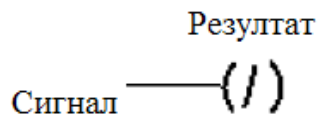
Тип на контактот	Симбол
Намотка	
Негирана намотка	
Намотка за сетирање	
Намотка за ресетирање	
Намотка со премин на позитивен раб	
Намотка со премин на негативен раб	
Намотка со премин на двата раба	

## 1. Намотка

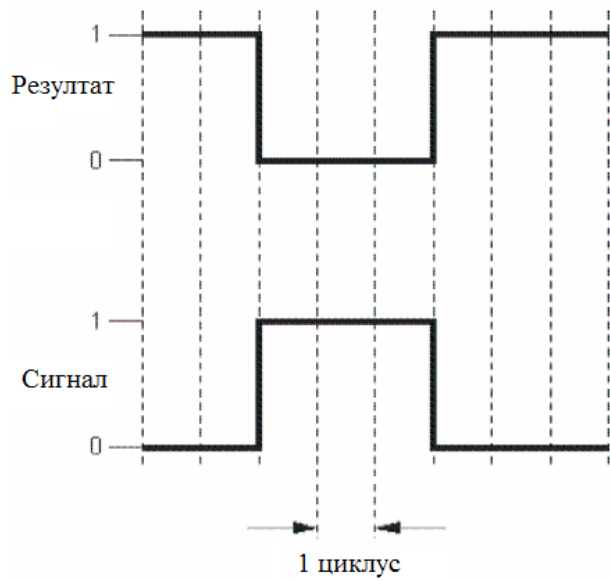


Намотката има состојба логичка 1 ако се исполнети условите во инструкцискиот дел.

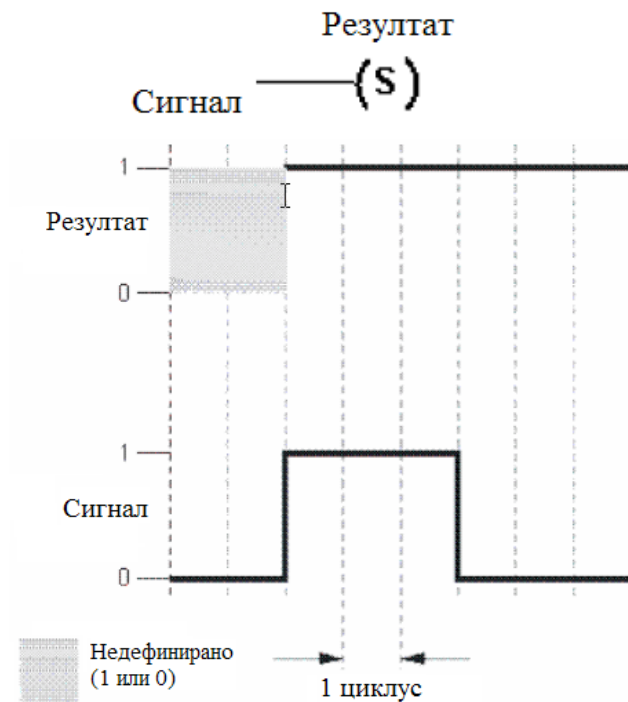
## 2. Негирана намотка



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



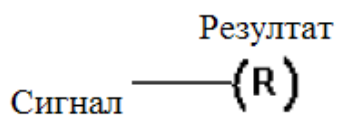
### 3. Намотка за сетирање



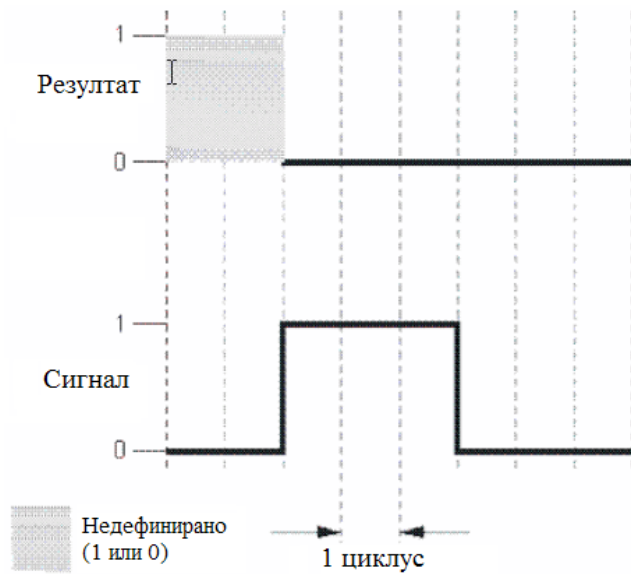
Намотката за сетирање ја сетира променливата во состојба 1 ако се исполнети условите.

Состојбата останува сетирана се додека променливата не се ресетира.

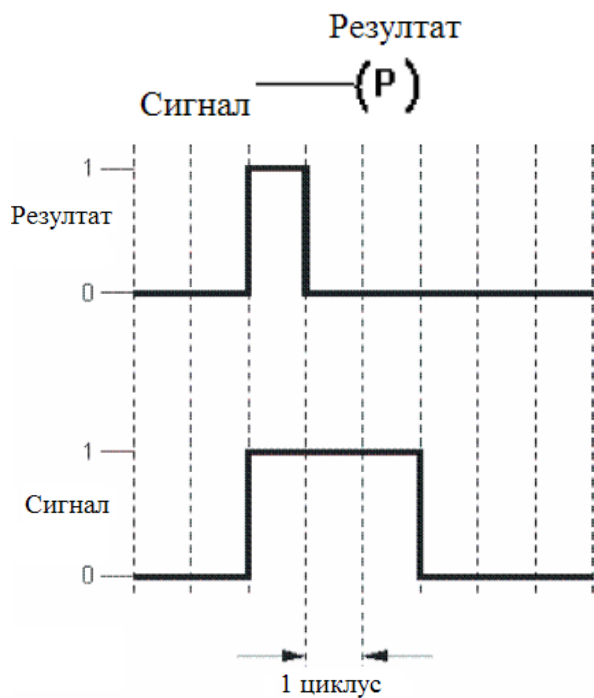
### 4. Намотка за ресетирање



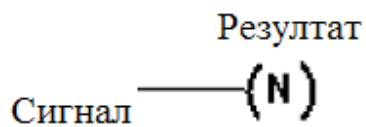
Ако се исполнети условите, намотката за ресетирање ја ресетира променливата, т.е. ја променува нејзината состојба во 0.



5. Намотка со премин на позитивен раб



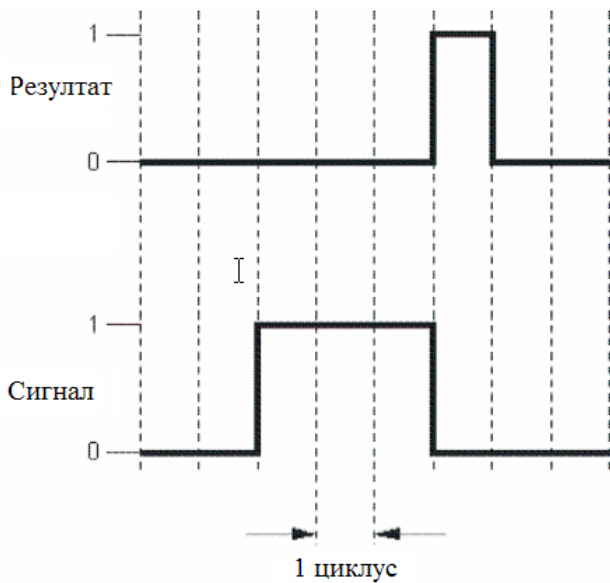
6. Намотка со премин на негативен раб



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.

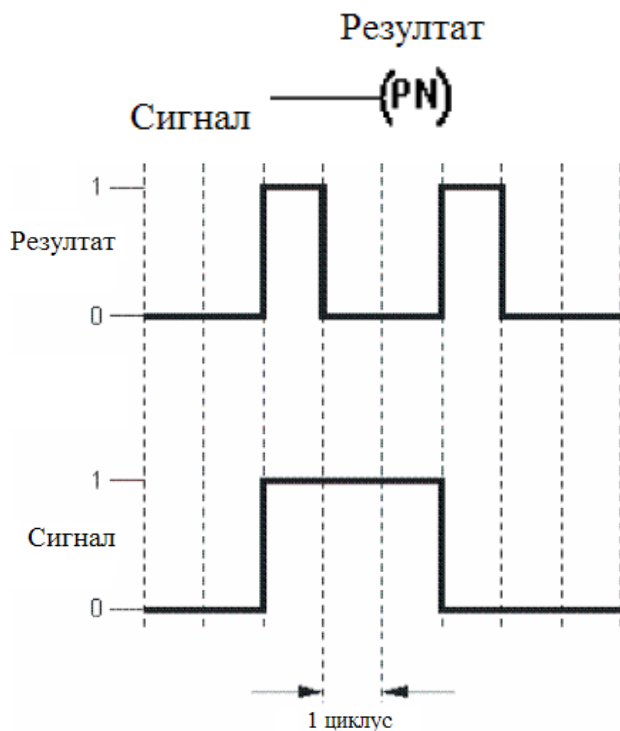
Намотката со негативен раб ја променува состојбата на променливата во логичка 1 за времетраење од 1 циклус, ако не се



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.

#### 4.2.5. Контролирање на текот на програмата

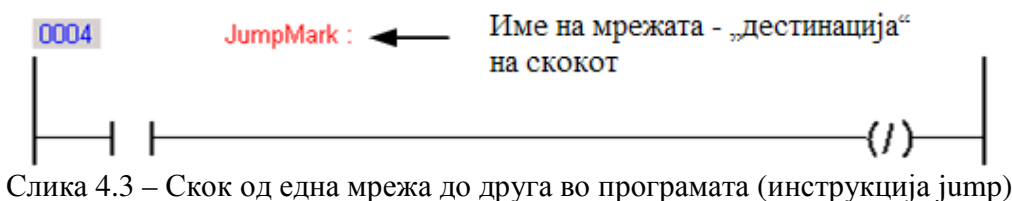
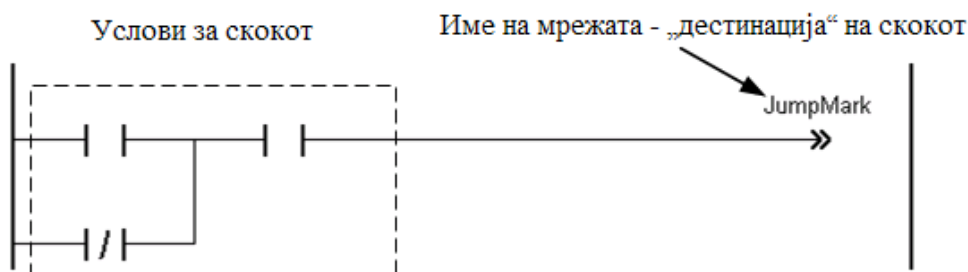
Текот на програмата се контролира со помош на две инструкции, **jump** и **return**. Идејата е оние мрежи од програмата кои не се потребни во дадениот момент да се прескокнат и да се изврши некој дел од програмата што се наоѓа под тие редови.



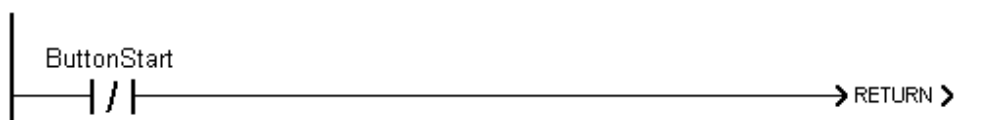
Кога се исполнети условите за инструкцијата jump, таа се извршува на тој начин што извршувањето на програмата прави „скок“ од местото на инструкцијата jump до онаа мрежа со специфично име, т.е. името на инструкцијата jump. На пример, ако инструкцијата jump е означена како „Jump Mark“, ќе се прескокнат 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 4.3).

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 4.4).



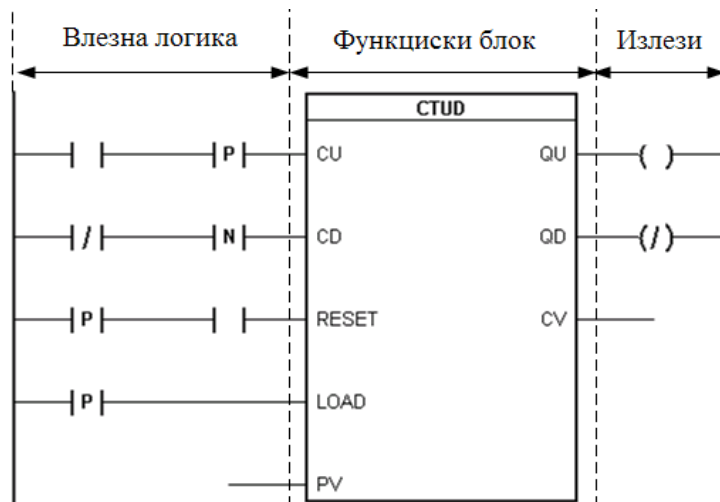
Слика 4.3 – Скок од една мрежа до друга во програмата (инструкција jump)




Слика 4.4 – Инструкција return

#### 4.2.6. 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).



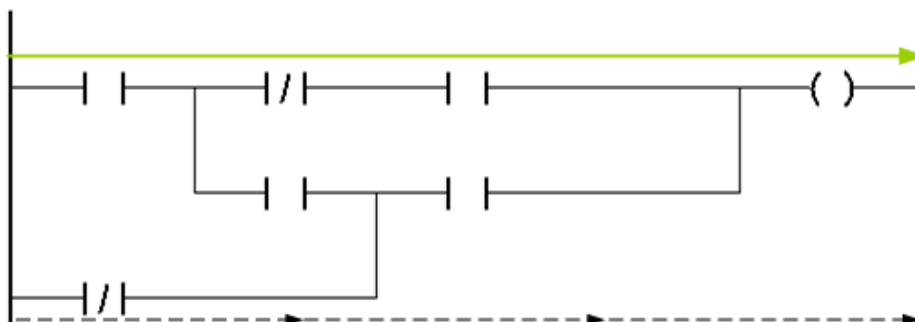
Слика 4.5 – Функциски блок во ледер дијаграм

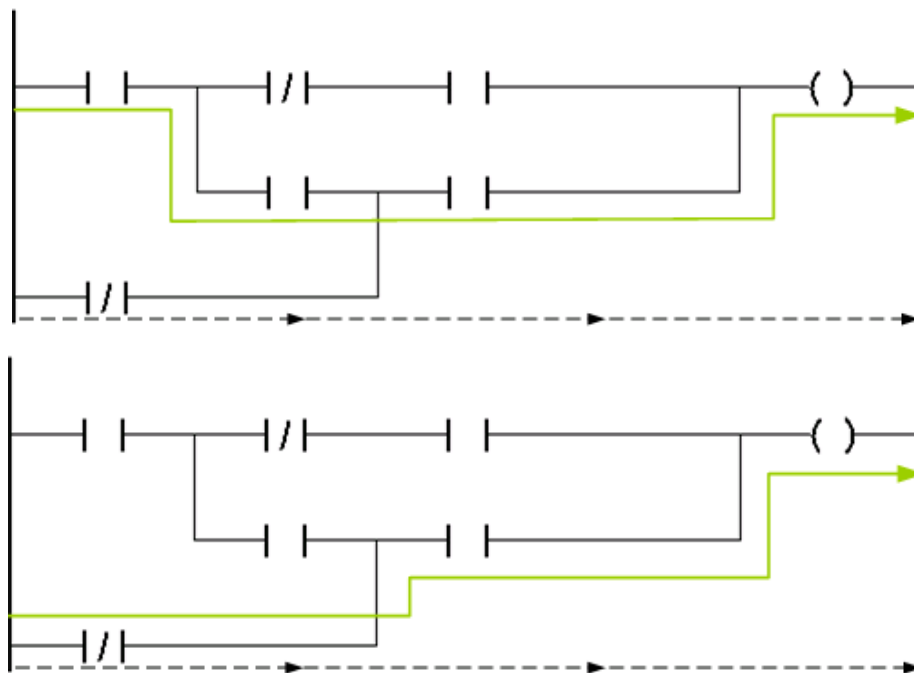
Функцискиот блок може да има и аналогни влезови и излези, поврзувајќи го со променливи на кои им се придружени некои од аналогните влезови и излези на модулите. Потребно е да курсерот да се постави во позиција на некоја од променливите на функцискиот блок (на пример, променливата CV од слика 4.5), и 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.

#### 4.2.7. 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).

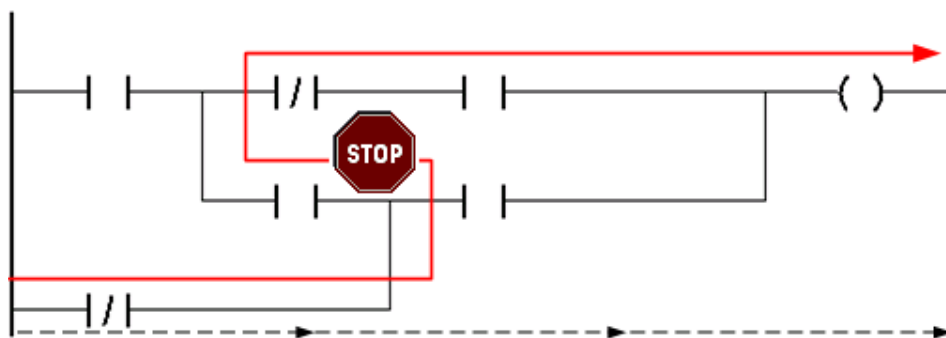
Во следнава мрежа има неколку можни начини на тек на логичкиот континуитет.





Слика 4.6 – Можни начини на тек на логичкиот континуитет

За разлика од жичано поврзаната релејна логика, тек на „струјата“ како што е прикажан на слика 4.7 не е возможна кај логиката на PLC контролерите.



Слика 4.7 – Не евозможен тек на логички континуитет од десно на лево

### 4.3. 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.

### 4.3.1. 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.

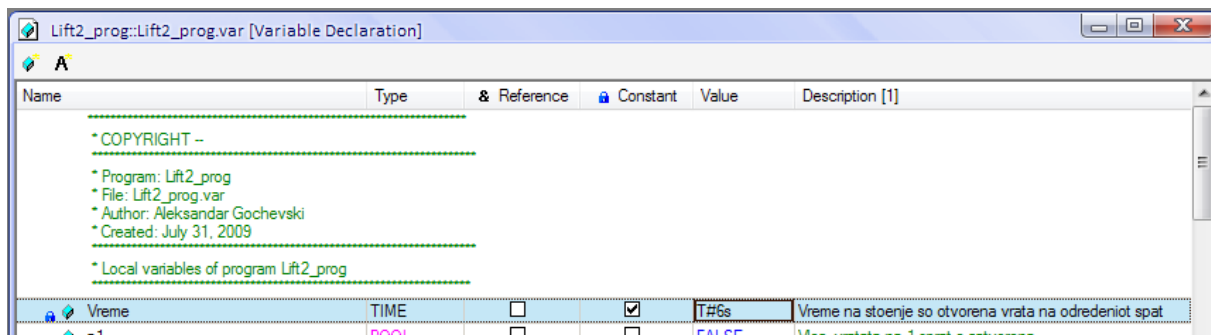
Бинарни	Целобројни ненегативни	Целобројни	Реални	Време, датум, знаковни променливи
BOOL	USINT	SINT	REAL	TIME
	UINT	INT	LREAL	DATE_AND_TIME
	UDINT	DINT		STRING

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


### 4.3.2. Декларирање на променливи и константи

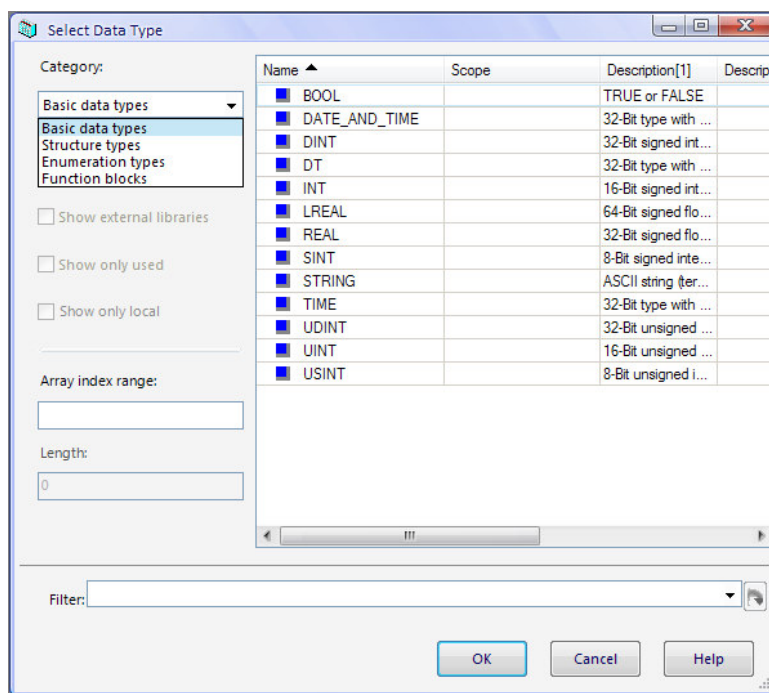
Променливите и константите се декларираат во фајлот со додавката (екстензија) **\*.var**. Начинот на декларирање на променливите беше објаснет во точката 3.2.2. При деларирањето на променливата, освен нејзиното име, треба да се специфицира и типот на

податокот што таа ќе го носи и дали таа е константа. На една променлива или константа може да се специфицира и одредена вредност (слика 4.8).



Слика 4.8 – Декларирање на константа и специфицирање на вредноста на променливата (константата)

Во колоната **Type** (слика 4.8) 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 прозорек како на слика 4.9.



Слика 4.9 – Избор на типот на променливата

Може да се избере помеѓу основни типови на променливи (Basic data types), структурирани типови на променливи (Structure types), набројиви 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 4.9), се избере Function blocks.

### 4.3.3. 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:

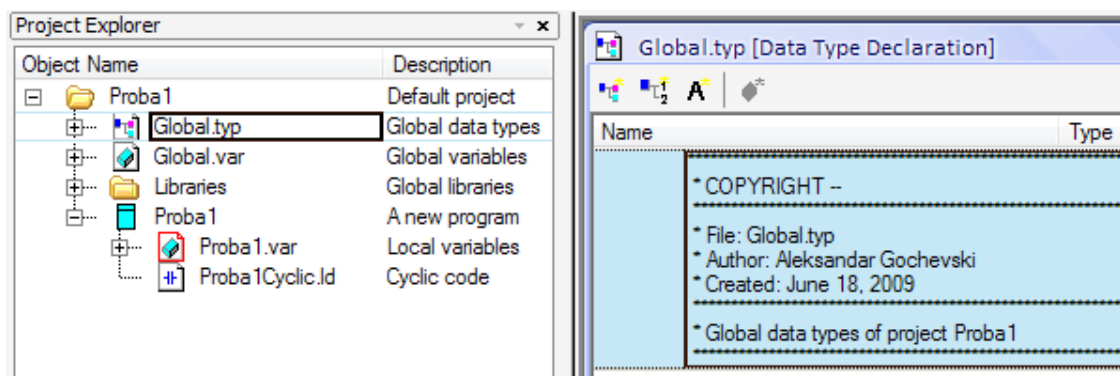
Voda  
Brasno  
Sol  
Kvasec

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.

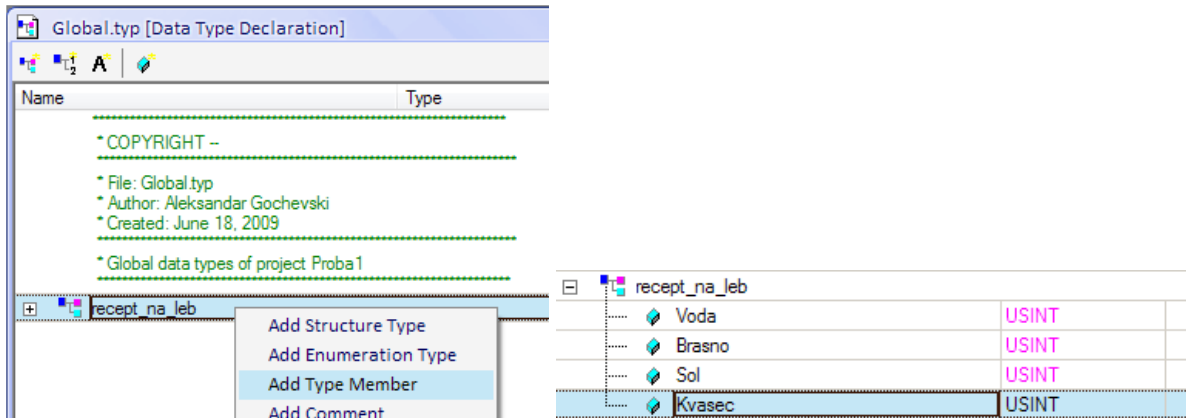
За да се креираат типови на податоци, треба да се отвори фајлот со екстензија \*.typ. Тоа е Global.typ (слика 4.10), се отвора со двоен клик и неговиот прозорец се појавува на десната страна од главниот прозорец.



Слика 4.10 – 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 4.11).

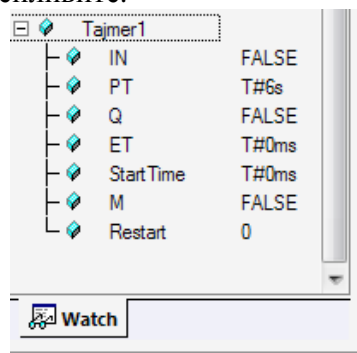


Слика 4.11

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

#### 4.3.4. 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 4.12 е даден пример на функцискиот блок TON поставен во прозорецот за следење на променливите.



Слика 4.12 – Променливата Tajmer1 на функцискиот блок TON во Watch прозорец

#### 4.3.5. Низи

Низите се променливи што содржат неколку елементи од ист тип на податоци. Контие 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:

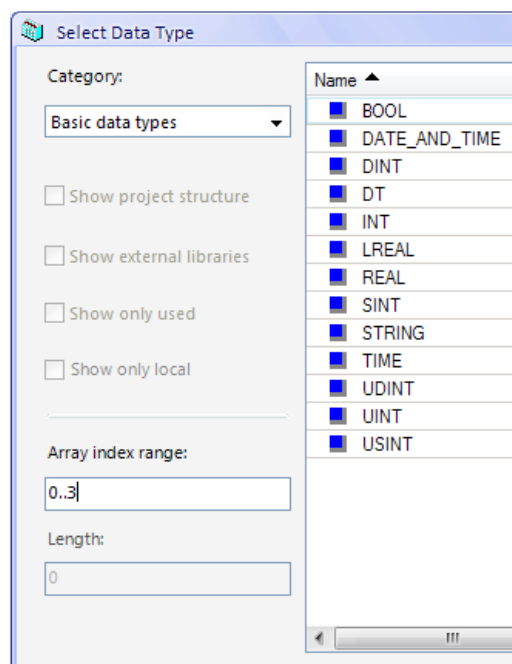
```
ArrayVariable [ArrayIndex]
```

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 4.13 – Декларирање на низа

#### 4.3.6. 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 4.14 – Глобални и локални променливи

#### 4.3.7. 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 4.8). 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 4.15).

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

Слика 4.15

**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 снемвање 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.

