Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство

Ниво студија: Основне академске студије

Предмет: Пројектовање информационог система и базе података

Број индекса: 565/2016

Милан Г. Ђекић

Пример примене ПЛЦ контролисаних уређаја, информационих система и база података у савременој индустрији

дипломски рад

Комисија за преглед и одбрану:

1. Др Александар Ђорђевић – ментор Датум одбране:_____

2._____ Оцена:_____

3._____

ПРИЛОГ 3: Образац пријаве завршног рада

Универзитет у Крагујевцу
Факултет инжењерских наука

Основне студије: **Рачунарска техника и софтверски инжењеринг**

Студијско подручје (усмерење): **Опште**

Име и презиме: **Милан Ђекић**

Број индекса: 565/2016

ПРИЈАВА ЗАВРШНОГ РАДА

Тема рада: Пример примене ПЛЦ контролисаних уређаја, информационих система и база података у савременој индустрији

Задатак: Задатак дипломског рада је опис функционисања и имплементација базе података за складиштења података који су добијени помоћу ПЛЦ контролисаних уређаја. У дипломском раду потребно је описати основе индустријских информационих система и база података, повод и сврху коришћења ПЛЦ контролера, а затим описати и поступак складиштења податка који се генеришу током рада ПЛЦ контролисаних уређаја. Коначно потребно је на практичном примеру утврдити и приказати значај примене информационог система у једном индустријском погону са дефинисамо базом података, која ће бити употребљена за складиштење података за даље планирање и управљање процесима који се заснивају на примени ПЛЦ контролисаних уређаја и на тај начин даље унапредити процесе и производњу.

Ментор:

Др Александар Ђорђевић

Крагујевац, 13.03.2019

Изјава о самосталној изради рада

Изјављујем да сам овај дипломски рад урадио самостално, служећи се стеченим знањем и искуством као и наведеном литературом.

Милан Г. Ђекић 565/2016

_____

(потпис)

Садржај

The summary

This documentation presents the functionality and implementation of information systems and databases for data storage obtained using PLC controlled devices. The importance of application of information system in one industrial plant and use of data from PLC devices is presented, which will later be used for further planning and management of processes based on the application of PLC controlled devices and thus improve processes and production.

Keywords: database, plc

Резиме

Овим радом описана је функционалност и имплементација информационих система и база података за складиштење података који су добијени помоћу ПЛЦ контролисаних уређаја. Представљен је значај примене информационог система у једном индустријском погону и употреба података са ПЛЦ уређаја који ће касније бити употребљени за даље планирање и управљање процесима који се заснивају на примени података са ПЛЦ контролисаних уређаја и на тај начин унапредити процесе и производњу.

Кључне речи: база података, плц.

1.Увод

Индустрија је данас суочена с новом генерацијом потпуно дигитализованих фабрика. Индустрија 4.0 као четврта индустријска револуција, већ траје у многим земљама. Таква индустрија обезбеђује опстанак индустрије и њен конкурентан развој у савременим условима. Овај дипломски рад се бави управо тиме. Покушава да имплементира делове индрустрије 4.0, повезивајући индустрију и производњу са процесуирањем података. У данашњем времену, подаци су најтраженија ствар. Овај пројекат и дипломски рад као такав има мисију да податке који се складиште на одређеном уређају употреби како би на основу различитих операција добили боље резултате у будућности као и увидели евентуалне пропусте у раду. Користићемо базу података и један систем који ће податке из те базе филтрирати и давати нам ближе информације о производном процесу, али и самом производу.. Подаци се памте тако да буду независни од програма који их користе, и структуирају се тако да је омогућен пораст базе. После сваке активности над базом, која представља логичку целину посла, стање базе мора бити конзистентно. Дакле, подаци у бази и односи међу њима морају задовољити унапред задате услове који осликавају део реалности моделиране подацима у бази.

У наставку рада биће описани процеси тестирања, складиштења података, и добијање података. Покушаћемо да овим дипломским радом унапредимо процесе производње, олакшамо операторима и инжењерима у индустрији њихове послове. Као резултат овог рада, биће представљен информациони систем који ће претраживати базу података коју ћемо из индустријског уређаја сместити у рачунар и програм и исту претраживати по одређеним параметрима.

2. Рад са базом података

2.1 Опис базе података

База података представља колекцију међусобно повезаних података који су организовани у табеле и друге структуре података, а користе једну или више апликација. Основна намена базе података је да буде репозиторијум (складиште) за податке. Подаци могу бити различитог типа, текстуални, нумерички, слике, аудио и видео записи и сл. Подаци у базама података су организовани у дводимензионалне табеле. Табела може да има више колона, где свака колона представља неку особину или атрибут. Врсте табеле чине конкретни подаци, односно конкренте вредности особина/атрибута неког објекта. На пример, једна табела може да садржи информације о ученицима. Колоне табеле могу да дефинишу име, презиме, годину рођења ученика, и сл. Врсте у таквој табели су ученици, тако да се свака врста односи на једног ученика. Које ће табеле да садржи база података зависи од проблема за који треба реализовати базу података.

На пример, база података се може одности на школу, па ће у том случају табеле бити о ученицима, наставницима, одељењима, и сл. Поступак избора и дефинисања табела за базу података је део процеса моделирања односно изградње модела података. Међусобна повезаност података је оно по чему се база података разликује у односу на фајл системе (датотеке) и програме за унакрсна израчунавања као што је *Excel*. Повезаност података обезбеђује значајне предности код претраживања када корисник може да на основу веза извуче много више података. На пример, ако постоји табела која чува податке о ученицима и табела са подацима о одељењима, веза између ученика и одељења може да обезбеди да одговарајућим захтевом (*SQL* упитом) прикажете све ученике жељеног одељења. База података садржи и тзв. метаподатке, односно податке о самој структури базе података. Метаподаци могу да се односе на имена табела, имена колона у свакој табели, на податке о корисницима података, као и разним помоћним структурама које обезбеђују брз прступ подацима (индекси).

2.2 *SQL* језик

За рад с релацијском базом података користе се упитни језици. Данашњи упитни језици који се користе су најчешће процедурални и непроцедурални. За релацијску базу података се користе непроцедурални језици. Постоји више непроцедуралних језика, а највише се користи *SQL* језик. Иако се назива упитним језиком, *SQL* има наредбе за свеобухватан рад с релацијском базом података, а то су:

- креирање таблице
- унос података у таблице
- брисање података из таблица
- наредбе за дефинисање базе података (*DDL-Data Definition Language*)
- наредбе за манипулацију подацима (*DML-Data Manipulation Language*)
- управљачке наредбе.

SQL (***Structured Query Language***) , тј, његов развој је започео 1974. године, када је објављен чланак *D.D.Chamberlin* и *R.F.Boyce* у којем они описују упитни језик за претраживање под називом *SEQUL*. Након тога 1975. им се придружује и *M.Hammer*, те они заједничко представљају концепт језика *SQUARE*. Убрзо након тога *SQUARE* мења назив у *SEQUEL2*, и тај језик је коришћен у развоју првог прототипа релацијског састава за управљање базама података који је имао назив *«System P»*, а касније тај језик мења име у *SQL*. *SQL* је релацијски потпун јер за свих 5 основних релацијских оператора постављао је семантички еквивалентне *SQL* наредбе. *SQL* језик је стандардизован од *ISO* (1986.г.) и *ANZI* (1986.године) института и данас се ради у већини релацијских система базе података. *SQL* омогућава прављење упита над више релација истовремено. Релацијска база података спада у савремене базе података уз објектни или димензијски модел. *SQL* је декларативан језик у којем корисник специфира само жељени резултат. *SQL* 1992 подупире само функције које крајњи корисник жели, али такође подупире и оне функције које обрађују апстрактне типове података.

Апстрактни типови података се данас могу додавати или брисати из система, без да то има икаквог утицаја на сам састав. Иако такав приступ повећава целокупну функционалност састава, сами састави нуде веома ограничену потпору за оптимисање операција *SQL*-стандардни упитни језик.

2.3 Основе система *RDBMS*

Савремени напредни системи *RDBMS* не само да складиште податке, већ и управљају њима тако што ограничавају податке који могу ући у систем, а такође омогућују узимање података из система. Ако вам је потребно само да сместите податке негде на сигурно, можете да користите било какав систем за складиштење података. Системи *RDBMS* вам омогућују да одете изнад самог складиштења података у домен дефинисања начина на који би ти подаци требало да изгледају или пословних правила за податке. Не треба мешати оно што се назива „пословним правилима за податке" са генерализованим пословним правилима која управљају целокупним системом (на пример, неко не може ништа да види док се не пријави на систем или аутоматско прилагођавање текућег периода у рачуноводственом систему првог у месецу). Ти типови правила у ствари се могу применити на било ком нивоу система (у данашње време то се обично примењује у средњем или клијентском слоју вишеслојног система). Уместо тога, оно по чему се овде говори су пословна правила која се посебно односе на податке. На пример, не може постојати поруџбина са негативном количином. Код система *RDBMS* та правила могу да се угради директно у интегритет саме базе података.

2.4 Преглед објеката базе података

Систем *RDBMS* као што је *SQL* сервер садржи велики број објеката. Код *SQL* сервера списак неких важнијих објеката базе података садржи ствари као што су:

- сама база података
- дневник трансакција
- склопови
- табеле
- групе датотека
- ускладиштене процедуре
- приказ
- извештаји
- функције које дефинише корисник
- корисници
- роле

2.4.1 Базе података као објекат

База података је у ствари објекат највишег нивоа који може да се референцира у оквиру датог *SQL* сервера. (Технички говорећи, и сам сервер се може сматрати објектом, али не из било какве реалне „програмерске" перспективе.) Већина других објеката *SQL* сервера, али не сви, изведени су од објекта базе података.

База података обично представља групу која садржи барем скуп објеката табела и најчешће друге објекте као што су ускладиштене процедуре и прикази који се односе на одређено груписање података сачуваних у табелама базе података.

Које типове табела чувамо у само једној бази података, а шта се налази у засебној бази података? Посматрајући једноставно рећи ћемо да се сви подаци за које се сматра да припадају само једном систему или да су значајно повезани, чувају у једној бази података. Систем *RDBMS* као што је *SQL* сервер може да има више корисничких база података на само једном серверу, а може да има и само једну. Број база података који може бити смештен на једном *SQL* серверу зависи од фактора као што су капацитет (снага *CPU, I/O* ограничења диска, меморија итд.), аутономија (желите да једна особа има право управљања сервером на којем се налази тај систем, а да неко други има администраторска права на другом серверу) или само број база података које ваша компанија или клијент има. Велики број сервера има само једну производну базу података, док их други могу имати и више. Треба имати на уму да код сваке верзије *SQL* сервера коју налазимо у употреби данас (*SQL* сервер 2000 је био већ пет година стар када је замењен), има могућност постојања већег броја примерака *SQL* сервера - укључујући и посебно пријављивање и посебна права управљања и то све на физички једном серверу.

Када се први пут покрене *SQL* сервер, почиње се са системским базама података:

- *мастер*
- *модел*
- *msdb*
- *tempdb*

2.4.2 Дневник трансакције

Сама датотека базе података није место на коме се дешава већина ствари. Иако се подаци дефинитивно читају одатле, свака измена која се прави не иде иницијално у саму базу података. Уместо тамо, они се серијски уписују у дневник трансакција. У неком каснијем тренутку база података задаје тренутак провере контролне тачке - у том тренутку времена све измене из дневника се преносе на саму датотеку базе података.

База података има уређење које се заснива на случајном приступу, али је сам дневник по својој природи серијски. Док случајан приступ датотеци базе података омогућује брз приступ, серијски приступ дневнику омогућује праћење ствари одговарајућим редоследом. Дневник акумулира измене за које се сматра да су урађене, а затим неколико измена одједном уписује у саму датотеку базе података. Да би имали функционалну базу података потребни су и датотека базе података и дневник трансакција.

2.4.3 Основни објекат базе података - табела

Базе података сачињавају многе ствари, али ни једна од њих није толико централна за саму садржину базе података као што је табела. Табела се може сматрати еквивалентом главној књизи рачуновође или радном листу у *Excelu*. Састоји се од такозваних података домена (колоне) и података ентитета (редови). Сами подаци из базе података чувају се у табелама.

Свака дефиниција табела садржи и метаподатке (описне информације о подацима) који описују природу података које би табела требало да садржи. Свака колона има свој скуп правила која одређују шта се може чувати у тој колони. Кршење правила у било којој колони може да доведе до тога да систем одбаци ред који желите да унесете или да одбије да ажурира постојећи ред или да спречи брисање реда.

2.4.4 Индекси

Индекс представља објекат који постоји само унутар радног окружења одређене табеле или приказа. Индекс функционише веома слично као и индекс на крају енциклопедије; постоји некаква вредност по којој се претражује (или кључна вредност), која је уређена на одређени начин и када се пронађе, добије се други кључ помоћу којег се могу пронаћи саме информације које су вам потребне.

Индекси обезбеђују начине који омогућавају да се убрза претраживање информација. Индекси спадају у две категорије:

Груписане - Само један овакав индекс може да постоји по табели. Ако је индекс груписани, то значи да је табела у којој се налази физички уређена у складу са тим индексом. Када би индексирали енциклопедију, кластерски индекс би био број стране; информације у енциклопедији су сачуване према редоследу бројева страна.

Негруписане – Може их бити више у свакој табели. Он се више уклапа у оно што се обично подразумева када се чује реч „индекс". Овај тип индекса указује на неку другу вредност која ће омогућити да се пронађу подаци. У енциклопедији, то би био индекс кључних речи на крају књиге.

Треба обратити пажњу на то да прикази који имају индексе или индексирани прикази - морају имати бар један груписани индекс да би уопште могли имати негруписане индексе.

2.4.5 Окидачи

Ограничење је још један објекат који постоји само унутар граница табеле. Ограничења су баш то што и њихово име говори, објекти који ограничавају податке у табели тако да испуњавају одређене критеријуме. Ограничења се на неки начин такмиче са окидачима као могућа решења за захтеве у вези са интегритетом података. Међутим, они не представљају исту ствар јер имају различите предности.

2.4.6 Групне датотеке

Према подразумеваној опцији, све табеле и све остало у вези са базом података (изузев дневника) чува се у једној датотеци. Та датотека је део нечега што се назива примарна група датотека. Међутим, то није све што се тиче уређења.

SQL сервер дозвољава дефинисање нешто више од 32000 секундарних датотека. Те секундарне датотеке могу се придружити примарној групи датотека или се могу направити као део једне секундарне групе датотека или више таквих група. Док постоји само једна примарна група датотека (и она се у свари назива „*Primary*"), могу постојати до 255 секундарних група датотека. Секундарна група датотека се прави као опција у оквиру команде *CREATE DATABASE* или *ALTER DATABASE*.

2.4.7 Дијаграми

Дијаграм базе података визуелно представља дизајн базе података укључујући различите табеле, имена колона у свакој табели и везе између табела. Постоји термин дијаграм ентитет-веза или *ERD*. На *ERD* дијаграму база података је подељена на два дела: ентитета (као што су „снабдевач" и „производ") и везе (као што су „набавка" и „куповина").

2.4.8 Прикази

Приказ је нешто налик на виртуелну табелу. У већини случајева, приказ се и користи као табела, али се од табеле разликује по томе што не садржи своје податке. Приказ представља унапред планирано мапирање и представљање података који се чувају у табелама. Тај план се чува у бази података у облику упита. Упит захтева податке из неколико колона (није неопходно да буду из свих) из једне или више табела. Враћени подаци можда морају, или не (зависно од дефиниције приказа) да испуњавају одређене критеријуме да би се појавили у том приказу. До *SQL* сервера 2000 основна сврха приказа била је контрола онога што корисник може да види. На то утичу две основне ствари: безбедност и лакоћа коришћења. Са приказима можете да контролишете оно што корисници виде, тако да ако постоји део табеле којем може да приступи само неколико корисника (на пример, детаљи у вези са платама), можете да направите приказ који садржи само колоне којима сви могу да приступе. Поред тога, приказ се може направити тако да корисник не мора да претражује скроз непотребне информације. Поред ових најосновнијих примена приказа, постоји могућност да се направи и такозвани индексирани приказ. Он је исти као и било који други приказ, изузев што у односу на тај приказ можемо да направимо индекс. [1]

То има неколико утицаја на перформансе (неки су позитивни, а неки негативни):

- Прикази који референцирају више табела извршавају се много брже ако је приказ индексиран зато што је спајање табела унапред направљено.
- Агрегирања која се обављају у приказу унапред се рачунају и чувају се као део индекса, то опет значи да се агрегација обавља у једном тренутку (када се унесе или ажурира неки ред), а затим се може читати из информација које садржи индекс.
- Уношење или брисање реда је спорије зато што и индекс у приказу мора одмах да се ажурира, ажурирања су такође спорија ако ажурирање утиче на колону која представља кључ у индексу.

2.4.9 Ускладиштене процедуре

Ускладиштене процедуре су историјски, па чак и у ери *.NET*-а, основа програмске функционалности код *SQL* сервера. Ускладиштене процедуре у ствари представљају низове исказа *Transact-SQL* (језик који се користи за задавање упита код *Microsoft SQL* сервера) спојених у једну логичку целину. Дозвољавају коришћење променљивих и параметара као и конструкција које омогућују избор и пролазак у циклусима. Ускладиштене процедуре нуде неколико предности у односу на слање појединачних исказа серверу због следећих разлога: Референцирају се коришћењем кратких имена, уместо дугачких низова текста, због чега је потребно мање мрежног саобраћаја како би се извршио код унутар ускладиштене процедуре. Оне су преоптимизоване и прекомпајлиране, чиме се штеди мала количина времена сваки пут када се ускладиштена процедура извршава. Учаурују процес, обично из безбедносних разлога или да би се сакрила сложеност базе података. Могу се позивати из других ускладиштених процедура, због чега се могу сматрати поново употребљивим у неком

ужем смислу. Поред тога, можете да користите било који *.NET* језик ако у ускладиштене процедуре желите да убаците програмске конструкције које се не налазе у матерњем језику *T-SQL*.[2]

2.4.10 Функције које дефинише корисник

Функције које дефинише корисник (*UDF*-ови) имају веома много сличности са ускладиштеним процедурама, а разликују се по томе што:

Могу да врате вредност чији тип може бити већина типова података који постоје код *SQL* сервера. Повратна вредност не може бити типа *text, image, cursor, timestamp*. Не могу имати „споредне ефекте". У основи, оне не могу даураде ништа изван домена функције, као што је мењање табела, слање електронских порука или мењање параметара система или параметара базе података.

Функције које дефинише корисник сличне су функцијама које бисте користили у стандардном програмском језику као што је *BC.NET* или *C++*. Можете да проследите више од једне променљиве и да добијете резултат на основу њих. Функције које дефинише корисник код *SQL* сервера разликују се од функција које можете наћи у многим процедуралним језицима али по томе да се све променљиве које се прослеђују функцији увек прослеђују по вредности. Међутим, постоје неке добре вести, а то су да можете вратити посебан тип података који представља табелу.

2.4.11 Корисници и роле

Ово двоје увек иду заједно. Корисници су прилично еквивалентни налозима. Укратко, овај објекат представља идентификатор потребан да би се неко пријавио на *SQL* сервер. Свако ко се пријављује на *SQL* сервер мора да се преслика у корисника (директно или индиректно зависно од безбедносног модела који се користи). Корисници припадају једној роли или већем броју рола. Права да се обаве одређени поступци код *SQL* сервера могу се доделити директно кориснику или роли којој припада један или више корисника.

2.4.12 Правила

Правила и ограничења ограничавају информације које могу да уђу у табелу. Ако ажурирани или убачени запис крши правило, убацивање или ажурирање ће се одбацити. Поред тога, правило може да се користи за дефинисање ограничења код кориснички дефинисаних типова података. За разлику од правила, ограничења нису баш самостални објекти, већ делови метаподатака који описују одређену табелу.

Правила се морају сматрати објектом који ту постоји само због компатибилности уназад и требало би их избегавати у новим апликацијама.

2.4.13 Подразумеване вредности

Постоје два типа подразумеваних вредности. Постоји подразумевана вредност која представља самосталан објекат и подразумевана вредност која у ствари и није објекат већ представља метаподатке који описују одређену колону у табели (слично као што имамо правила, која представљају објекте, и ограничења, која нису објекти већ метаподаци). Имају исту намену. Ако приликом убацивања новог реда не обезбедите вредност за неку колону, а колона има дефинисану подразумевану вредност, аутоматски ће се убацити вредност која је дефинисана као подразумевана.

2.4.14 Типови података које дефинише корисник

Типови података које дефинише корисник представљају проширења за типове података које дефинише систем. Почевши од ове верзије *SQL* сервера, могућности у овој области су скоро бескрајне. Иако су *SQL* сервер 2000 и претходне верзије имали појам типова података које дефинише корисник, ти типови података били су ограничени на различито филтрирање постојећих типова података. Код *SQL* сервера 2005 имате могућност да везујете *.NET* склопове за ваше типове података, што значи да можете имати тип података који чува (са разлогом) све што можете сачувати у *.NET* објекту.

2.4.15 Подаци типа *NULL*

Шта ако имате ред који не садржи никакве податке у одређеној колони, то јест, шта ако једноставно не знате ту вредност? На пример, рецимо да постоји запис који покушава да чува информације о раду компаније током дате године. Сада замислите да је једно од поља проценат пораста у односу на претходну годину, али немате записе за годину пре првог записа у бази података. Можете доћи у искушење да унесете нуле у колону. Међутим, да ли ће то обезбедити праве информације? Они који не знају детаље могу помислити да та нула означава да сте имали проценат раста од нула посто, а не чињеницу да једноставно не знате вредност за посматрану годину.

Вредности које су неодређене означавају се као вредности *NULL*. Веома је тешко дефинисати вредност *NULL* јер значи да не знате која је вредност. Могла би бити 1; могла би бити 347; могла би бити -294 и то је све што знамо. Укратко, значи да је вредност недефинисана или можда неприменљива.

3. Основне наредбе *SQL*-а

Основна наредба у *SQL*-у је *SELECT* израз *FROM* име таблице, у преводу ИЗАБЕРИ израз ИЗ име таблице. У наредби реч ИЗРАЗ замењује се са именима редова које је потребно приказати или у случају да је потребно видети све са *. Поред имена реда може се написати ново име реда под којим ће се видети у извештају, али у таблици остаје све по старом.

Следећа *SELECT* наредба се разликује код *SQL*-а и *SAP*-а. Као што је приказано у претходној наредби да сваки ред има своје име, али некад може бити потребно да се у неком извештају споје два реда таблице у један ред извештаја. Спајање се врши у *SQL* -у са знаком +, док се иста радња у *SAP*-у ради са знаком назива пипе (*Alt Gr + W*). Можемо још споменути и кључну реч *TOP x*, која нам даје само првих *x* редова, што је корисно при раду са великим таблицама када често морамо проверавати резултате наших израза. Постоји још једна могућност коришћења наредбе *SELECT* без додатка *FROM* јер тај податак не мора бити унешен у таблицу. Следећи користан додатак наредби *SELECT* је захтев *WHERE* који показује који се редови могу видети јер у већини претходних примера имамо приказане све редове.

Наредни корак уношења података би био употреба логичких оператора тако да је могуће још више проширити захтеве за претраживање. Логички оператори су *AND, OR* и *NOT*. Код сложенијих израза би требало водити рачуна о приоритетима. Хијерархија примене израза гласи:

- Заграда ( ),
- Дељење / и множење *,
- Сабирање + и одузимање -,
- *NOT* (не),
- *AND* (и),
- *OR* (или).

Ако је потребно дефинисати у захтеву распон имамо две могућности. Једна је да радимо са операторима упоређивања (<, =, >, ...), а друга је да користимо кључну реч *BETWEEN*. Међутим, уколико је потребно наћи податак којем је захтев да се поклапа са једном од вредности у листи користимо кључну реч *IN*. Некад се може догодити да тражимо, на пример, неко име које почиње као ... или телефонски број. Тако задате захтеве је мало теже добити кроз прошле наредбе или операторе, па за то служи кључна реч *LIKE*. Оператор *IS NULL* нам даје информацију, на пример, за коју особу из таблице немамо податке. Уз коришћење ове кључне речи може се добити резултат који је читљивији, односно можемо резултат сортирати по неком редоследу.

Међутим, кад постоје само бројеви или само слова јасно је како ће бити сортирани ти низови, али кад је све то измешано поставља се питање, како ће сад то бити сортирано? Ту можемо позвати у помоћ једну процедуру у *MS SQL*-у која ће нам дати редослед низа по којем се врши сортирање, а то је *EXEC SP_HELPSORT.*

4. Складиште података

Складиште података је скуп података на којем се темељи систем подршке у одлучивању. Из сврхе произилази да складиште података треба да подацима потпуно "покрије" једно или више пословних подручја (нпр. набавке, продаје), да подаци у складишту требају да буду свеобухватни. Подаци морају да обухвате дужи временски период (пет, десет или више година), јер су временске анализе пословно врло значајне. Оријентисаност према пословним анализама не захтева од складишта да се подаци брзо ажурирају као у бази података. Мање складиште података које обухвата податке само једног пословног подручја назива се подручним складиштем података (*Data mart*).

Складиште података намењено је менаџерима, али и свима који у свом послу обављају различите аналитичке задатке, као што су послови праћења и извештавања, који се темеље на примени различитих пословних правила.

Подаци у складишту су стабилни и кад се једном сместе у складишту по правилу се не мењају. Тиме се омогућује да менаџемент или свако ко користи складиште података може бити сигуран да ће добити једнак одговор независно од времена или учесталости постављања упита. Поступак складиштења података представља континуалан процес планирања, грађења и прикупљања података из различитих извора, његовог коришћења, одржавања, управљања и сталног унапређења. Међу многим корацима у том комплексном континуалном процесу битно је нагласити важност поседовања визије о томе што се жели постићи креирањем складишта података. Једна од улога складишта је пример развоја и коришћење знања заснованог на подацима *(data-based knowledge).*

Главни циљ складишта података је ослободити информације које су "закључане" у базама података и "помешати" их са информацијама из осталих, по правилу спољних извора података. Велике организације данас све више траже додатне податке из спољашњих извора, као што су на пример подаци о конкуренцији, демографски трендови, продајни трендови и сл.

Да би складиште података могло да испуни циљ и сврху свог постојања, мора пре свега да испуни одређене предуслове:

1. Мора осигуравати приступ свим запосленим у предузећу, а не само менаџерима, значи може служити великом броју људи. Тај приступ мора бити поуздан, брз и једноставан.
2. Складиште треба садржати велику количину детаљних података. То значи да све пословне трансакције релевантне за доношење пословних одлука, које су настале у процесима предузећа морају бити евидентиране у складишту података. Унесени подаци требају бити конзистентни, нпр. ако је са два различита места у различито време постављен једнак упит и резултат тих упита мора бити исти.
3. Освежавање, односно ажурирање новим подацима треба бити континуалан процес, по могућности треба се одвијати у стварном времену практично одмах након што се неки пословни догађај одиграо или одмах по завршетку неког процеса.
4. Мора бити увек расположиво и обликовано на начин да може послужити свакој ситуацији коју није увек могуће унапред предвидети.

5. Треба предвидети могућност издвајања и међусобног повезивања података у смислу добијања свих мера и показатеља пословања у подузећу.
6. Подаци у складишту који се скупљају из различитих извора, контролишу се уз осигурање квалитета и само такви су доступни корисницима. Лоши улазни подаци не могу давати добре излазне податке.
7. Мора бити проширило да би могло пратити стратегију проширења пословања предузећа.
8. И на крају, мора да задовољи одговарајуће мере заштите тајности осетљивих података што се постиже спровођењем ригорозних мера чувања тајности.

5. ПЛЦ програмабилни логички контролер

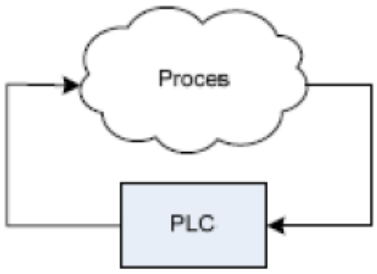
5.1 Увод

Индустријски контролер или ПЛЦ је дигитални електронски уређај који поседује програмабилну меморију за смештање инструкција којима се реализују специфичне функције, као што су логичке и артиметичке операције, редоследнно извршавање различитих акција, одмеравање временскиј интервала, пребројавање догађаја итд. Све у циљу управљања различитим машинама и процесима путем дигиталниј и/или аналогних улазно-излазних јединица.ПЛЦ је наменски електронски уређај заснован на микропроцесору, који је у могућности да обавља бројне типове управљачких функција различитог нивоа сложености. ПЛЦ индустријски рачунар чији су хардвер и софтвер посебно прилагођени раду у индустријским условима а који се лако може програмирати и уграђивати у нове и постојеће индустријске системе.

•**P**LC (Programmable) - означава могућност програмирања. Програм рада се припремаунапред и смешта у перманенту меморију ПЛЦ контролера. ПЛЦ програм се развија у језику лествичастих (ладдер) дијаграма, који је настао по угледу на тзв. релејне шеме.

• P**L**C (Logic) – означава могућност обављања логичких (Булових) функција је једна од главнихфункционалних карактеристика ПЛЦ контролера. ПЛЦ генерише дискретне (дигиталне) излазнесигнале у функцији (логичкој) дискретних улазних сигнала - карактеристично за првобитне типовеПЛЦ контролера. Савремени ПЛЦ-ови, поред логичких могу додатно да обављају аритметичкеоперације, одмеравају временске интервале, пребројавају догађаје, а прихватају и генеришу,поред дискретних, и аналогне сигнале.

• PL**C** (Controller) - главну примену ПЛЦ-ови налазе у индустрији (производној) где секористе за аутоматско управљање процесима - прати кључне параметре процеса (посредствомприкључених сензора и давача, и сходно меморисаном програму, генерише побуду којомделује на процес (посредством актуатора). [5]



Слика 1. Управљачка јединица процеса

На слици 1. видимо приказ где ПЛЦ уређај прати кључне параметре процеса.

5.2 ПЛЦ vs ПЦ

ПЛЦ контролер се разликује од рачунарског система опште намене по томе што не поседује спољну меморију (дискове), као и низ стандардне улазно/излазне опреме (миш, тастатура, ...). Поред тога, оперативни систем ПЛЦ-а је једноставнији и пружа компаративно мање могућности од рачунара опште намене. Заправо, ПЛЦ је конципиран и пројектован за један релативно узак и јасно дефинисан обим послова везаних за надзор и управљање појединим уређајима, машинама и процесима, што је резултовало у његовој изузетној ефикасности и једноставности.

5.3 .Историјат ПЛЦ-ова

Настанак ПЛЦ-ова се везује за касне 60’ и ране 70’ године прошлог века, а настали су на бази конвенционалних рачунара тог времена. Њихова првобитна примена била је у аутомобилској индустрији, а са циљем да се скрати време застоја у производњи услед промене производног процеса. Припрема погона за производњу новог модела аутомобила трајала је месецима, што је захтевало преповезивање панела и ормара препуних проводника, релеа, тајмера, склопки и других, углавном електро-механичких компоненти, помоћу којих су се, у то време, реализовале управљачке функције. Увођење ПЛЦ-а, омогућило је да се репрограмирање управљачке логике обави ″преко тастатуре″ и да се, уз минимална додатна преповезивања, време застоја производње скрати до тек неколико дана.

Главни проблем са рачунарима/ПЛЦ-овима из 70’ година односио се управно на њихово репрограмирање. Програми су били писани на ниском нивоу (асемблер), а тиме и компликовани, а њих су могли да састављају само високо-стручни и искусни програмери. Касних 70‘ година учињен је известан напредак у погледу поједностављења процедуре програмирања. 1972. године појављују се микропроцесори, који, захваљујући повољном односу перформансе/цена брзо налазе широку примену у многим областима електронике, па и индустријске аутоматизације. Од тог времена, па до данас, ПЛЦ-ови се непрестано усавршавају, како у погледу перформанси (моћи обраде), софтверске подршке, могућности спрезања са најразличитијим уређајима, тако у погледу начина и процедура програмирања, као и језика који се користе за њихово програмирање. На тај начин, ПЛЦ-ови су постали ″разумљивији″ широј популацији индустријских инжењера. 80‘ година долази до експоненцијалног раста тржишта ПЛЦ-ова, а њихова примена се шири ван индустријских погона, и то на системе као што су електро-енергетска постројења, системи аутоматизације стамбених/пословних објеката, системи обезбеђења и надзора итд. Данас, ПЛЦ-ови се све више примењују и у областима као што је медицинска опрема и уређаји, кућни апарати и сл.

Први ПЛЦ контролери су били једноставни уређаји за он/офф управљање и превасходно су се користили за замену застареле релејне технике. Међутим, такви ПЛЦ контролери нису могли да обезбеде сложеније управљање, као што је управљање температуром, притиском, позицијом. У међувремену, произвођачи ПЛЦ контролера развили су и уградили у ПЛЦ контролере бројна побољшања и функционална унапређења. Савремени ПЛЦ контролери имају могућност обављање изузетно

сложених задатака као што је управљање прецизним позиционирањем и управљање сложеним технолошким процесима. Такође, брзина рада ПЛЦ контролера је значајно повећана, као и лакоћа програмирања. Развијени су бројни модули специјалне намене за примене као што је радио комуникација, визија или чак препознавање говорних команди. Сматра се да инжењер са знањем релејне логике може овладати основним ПЛЦ функцијама у року од неколико сати. Слично важи и за инжењере са знањем дигиталне електронике. За овладавање сложенијим функијама и могућностима ПЛЦ-а довољно је неколико недеља или један курс. Данас постоји велики број различитих типова ПЛЦ контролера који се разликују по величини, изгледу и моћи обраде, почев од малих јединица са малим и ограниченим бројем улаза и излаза до великих, модуларних система који се могу конфигурисати за рад са више стотина или чак хиљада улаза/излаза. На Сл. 4-2 је приказан изглед ПЛЦ контролера из фамилије Аллен Брадлеу СЛЦ 500 Модулар Цонтроллерс. ПЛЦ се састоји из шасије (рацк) која има одређени број слотова у које се стављају поједини модули. Први два слота у шасији заузимају уређај за напајање и процесорски модул, док је распоред модула у преосталим слотовима произвољан. У зависности од броја модула, ПЛЦ може имати и више од једне шасије. Свака шасија има сопствено напајање, док се процесорски модул налази само у првој шасији. [6]

Слика 2. Изглед компоненти модуларног ПЛЦ система.

Приказ модуларних ПЛЦ уређаја, који се називају и шински, јер модули могу да се каче на шину приликом монтирања.

5.4 Предности ПЛЦ-ова

Флексибилност. У прошлости, свака електрично-управљана машина за производњу захтевала је своју сопствену управљачку јединицу; у погону са 15 машина, постојало је 15 различитих, наменски пројектованих, управљачких јединица. Данас је могуће исти модел ПЛЦ-а користити за управљање било којом од 15 машина. Уз то, вероватно неће бити потребе за 15 ПЛЦ-ова, јер један ПЛЦ лако може да опслужи више независних машина, тако што ће, конкурентно, за сваку прикључену машину извршавати посебан, наменски програм.

Лака промена програма и корекција грешака. Код традиционалних, релејних панела, свака промена програма рада захтевала је значајан утрошак времена ради преповезивање панела и уређаја. С друге стране, код ПЛЦ-а, промена програма је лака и брза. Нови програм се преко тастатуре или на неки други начин учитава у ПЛЦ, а преповезивање обично није потребно, тако да целокупна активност не траје дуже од неколико минута. Такође, уочене неправилности у раду система, које су последица грешке у програму могу се лако и брзо исправити.наменски програм.

Велики број контакта. Функција релејне управљачке јединице се реализује повезивањем релеа (контакта) који имају улогу електро-механичких логичких кола или флип-флопова. Број релеа у једном релејном панелу је ограничен (нпр. физичким димензијама панела), што ограничава и сложеност функције која се може реализовати. Може се десити да у случају промене ″програма″ рада упављачке једице, број распложивих релеа више није довољан да би се реализовала нова функција. У таквим случајевима, неопходно је уградити додатне релее или релејне блокове, што може бити дуготрајна операција скопчана с неизбежним механичким интервенцијама. С друге стране, ПЛЦ може реализовати функције чија је сложеност ограничена једино расположивом меморијом.

Ниска цена. Упоредо с напретком технологије, имплементационе могућности ПЛЦ-ова непрестано расту. Данас је могуће, по цени испод 100$, набавити ПЛЦ са огромним бројем интерно-расположивих ″виртуелних″ релеја, тајмера, бројача, секвенцера и других функција.

Могућност пробног рада. Рад ПЛЦ-а се може испитати у лабораторији, пре уградње у производни погон. Програм се пише, тестира, анализира и, ако је неопходно, модификује све до тренутка када се процени да су све захтеване функције коректно реализоване. Тек тада се програм преноси у ПЛЦ који се инсталира (или је већ инсталиран) у производни погон. На овај начин, постиже се велика уштеда скупог „фабричког“ времена (нема застоја у производњи). Насупрот томе, тестирање конвенционалних релејних система се може обавити само у фабричкој хали, што може бити веома временски нерационално.

Могућност визуелног праћења рада. Рад ПЛЦ-а се може директно пратити на екрану монитора - на погодан начин се у графичком облику приказују стања улаза и излаза ПЛЦ-а уз „осветљено“ приказивање логичких путања које су тренутно активне и исписивање обавештења о евентуалном неисправном раду система или о настанку неких изузетних ситуација. На тај начин, свака критична ситуација се може уочити истог тренутка када се и десила, што олакшава проналажење насталих кварова. Наиме, код напредних ПЛЦ система, постоји могућност програмирања порука за свако могуће неисправно понашање, које ће се приказати оног тренутка када се таква ситуација детектује од стране ПЛЦ-а, (као нпр. „МОТОР #7 је преоптерећен“).а временски нерационално.

Брзина рада. У поређењу са релеима који су, с обзиром на своју електро-механичку природу, спори, брзина извршења ПЛЦ програма је веома велика. Брзина рада ПЛЦ-а одређена је трајањем тзв. скен циклуса, што је реда милисекунди. Другим речима, време које протекне од тренутка када се промени стање улаза ПЛЦ-а до тренутка када

ПЛЦ-а реагује постављајући своје излазе типично није дуже од неколико до максимално неколико десетина милисекунди.

Поузданост и лакоћа одржавања. Полупроводничке компоненте, од којих је ПЛЦ сачињен, су, генерално, поузданије од механичких система или релеа и тајмера. Уз то, за реализацију ПЛЦ-ови се користе компоненте са веома високим фактором поузданости. Из тог разлога, трошкови одржавања управљачких система заснованих на ПЛЦ-у су нижи, а време застоја краће..

Ледер програмирање. За програмирање ПЛЦ контролера користи се језик лествичастих логичких дијаграма (или ледер дијаграма - ладдер диаграм), који је већ дуги низ година у употреби у индустрији при пројектовању логичких и секвенцијалних релејних управљачких јединица. Овај језик користи графичку нотацију која је по визуелном изгледу и логици рада слична дијаграмима релејних шема и због тога је лако разумљив индустријским инжењерима. Другим речима, индустријски инжењери не морају бити експерти за програмирање да би у својим системима користили ПЛЦ-ове.

Поузданост и лакоћа одржавања. Полупроводничке компоненте, од којих је ПЛЦ сачињен, су, генерално, поузданије од механичких система или релеа и тајмера. Уз то, за реализацију ПЛЦ-ови се користе компоненте са веома високим фактором поузданости. Из тог разлога, трошкови одржавања управљачких система заснованих на ПЛЦ-у су нижи, а време застоја краће.

Једноставност наручивања компоненти управљачког система. ПЛЦ је један уређај. Када наручени ПЛЦ стигне у индустријски погон, сви бројачи, релеи, и друге ″виртуелне″ компоненте ″садржане″ у ПЛЦ-у су такође стигле. Ситуација је потпуно другачија када се пројектује релејни панел. На пример, потребно је 20 различитих типова релеа и тајмера који се набављају од 12 различитих добављача, са различитим роковима испоруке и потенцијалним проблемима са доступношћу тражених компоненти. Ако пројектант пропусти да наручи само једну компоненту, целокупан пројекат се одлаже за време испоруке те компоненте. Код ПЛЦ-ова увек постоји „један реле више“ - под условом да је наручен ПЛЦ са извесном резервом у моћи израчунавања.

Документација. Ледер дијаграми, као графички прикази, су у тој мери само-дескриптивни да обично није неопходна нека додатна документација која би употпуњавала опис рада ПЛЦ-а и начина на који су реализоване његове функције. Ледер дијаграм се увек може одштампати, а пошто се исти дијаграм користи и као програм, не постоји опасност да документација буде неажурна, што је често случај са дијаграмима и шемама релејних панела (када инжењер након учињене интервенције не унесе измену у релејну шему).

Безбедност. Програм ПЛЦ-а се не може променити пре него што је ПЛЦ „откључан“ (сваки ПЛЦ има прекидач с кључем - без кључа није могуће мењати програм).

Могућност репрограмирања. Особина брзог репрограмирања ПЛЦ-а отвара могућност за постизање неке врсте адаптивног производног процеса, где се програм рада мења сходно карактеристикама сваког појединачног производа или варијацијама у процесу производње.[5]

5.5 Недостаци ПЛЦ-ова

Нова технологија. Показало се да је тешко променити начин размишљања индустријских инжењера са релејне логике на ПЛЦ концепт. Међутим, захваљујући све широј примени, не само у домовима и канцеларијама већ све више и у фабричким халама, рачунари постају и средство за постизање веће продуктивности производње.

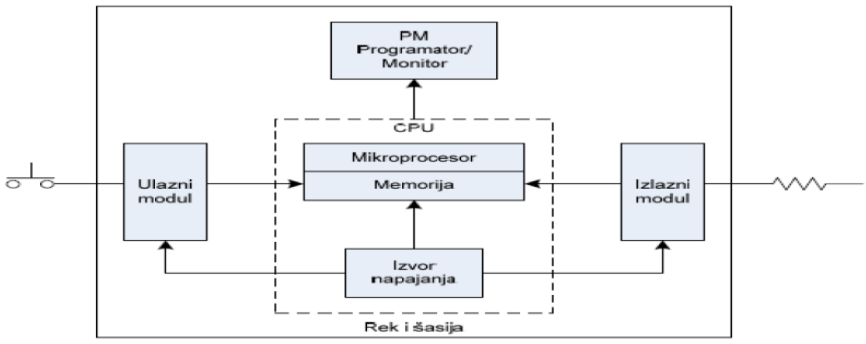
Апликације са фиксним програмом. Поједине апликације засноване су на само једној функцији која се веома ретко или никада не мења. У оваквим случајевима, замена постојеће опреме ПЛЦ-ом не доноси велики добитак, јер се њихова главна особеност – могућност репрограмирања – практично не користи. ПЛЦ је најбоље решење када су неопходне периодичне промене у начину рада.

Услови рада. Екстремним услови рада, као што су: висока температура, влажност, вибрације, електричне сметње, а који су карактеристични за поједине производне процесе, могу утицати на рад ПЛЦ-а и ограничити његову примену и/или животни век.

Безбедност у раду. Код релејних система увек постоји тзв. Стоп прекидач, којим се у било ком моменту може тренутно прекинути рад система (искључењем напајања). При томе, релејни систем се аутоматски не ресетује када се напајање укључи, већ задржава стање у коме је био када је напајање искључено. Овакво понашање се свакако може програмски остварити и код ПЛЦ-а, тако што ће се Стоп прекидач повезати на један од улаза ПЛЦ-а. Међутим, овакво решење није безбедно (ако ПЛЦ откаже, прекидач Стоп губи функцију!) Овај недостатак се може превазићи уградњом безбедоносних релеа на излазима ПЛЦ-а.[6]

5.6 ПЛЦ систем

На слици 3. су приказане четири основне јединице сваког ПЛЦ система као и начин на који су међусобно повезане.



Слика 3. ПЛЦ систем.

1. Централна процесоркса јединица (ЦПУ) или логичка јединица. Представља „мозак“ система, а састоји се из следеће три пођединице:

- Микропроцесор

- Меморија – за чување системског софтвера и корисничког програма

- Извор напајања – обезбеђује напајање микропроцесора, меморије, улазног и излазног модула.

2. Програматор/Монитор (ПМ). ПМ је уређај који се користи за комуникацију са ПЛЦ-ом. Примери ПМ-ова су: ручни терминали, индустријски терминали и персонални рачунари. Са једне стране, према оператеру, ови уређаји поседују екран и тастатуру, док са друге, према ПЛЦ-у, одговарајући комуникациони интерфејс за пренос програма, података, статусних информација ка/из ПЛЦ-а.

3. У/И модули. Улазни модул поседује терминале (прикључне тачке) на које се доводе електрични сигнали које генеришу сензори или преварачи. Излазни модул поседује терминале преко који ПЛЦ шаље излазне сигнале којима побуђује релее, соленоиде, моторе, дисплеје и друге излазне уређаје или актуаторе.

4. Рекови и шасије. Делови ПЛЦ система ЦПУ, ПМ и У/И модули смештају се у металне ормаре– тзв. рекове.[8]

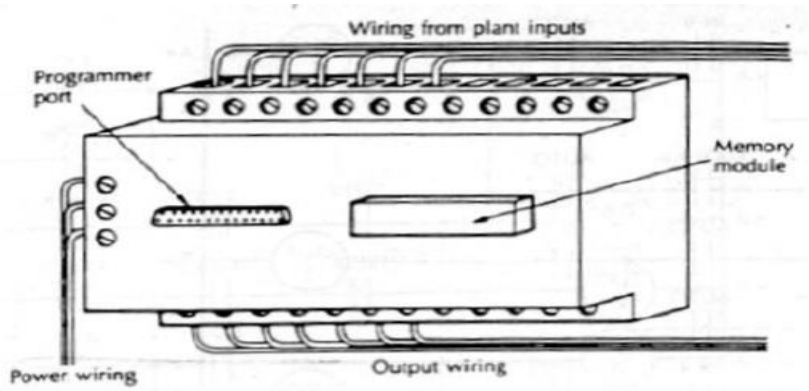
5.7 Конструкција ПЛЦ-а

Разликују се два основна начина конструкције ПЛЦ контролера:

(а) компактни ПЛЦ контролери и

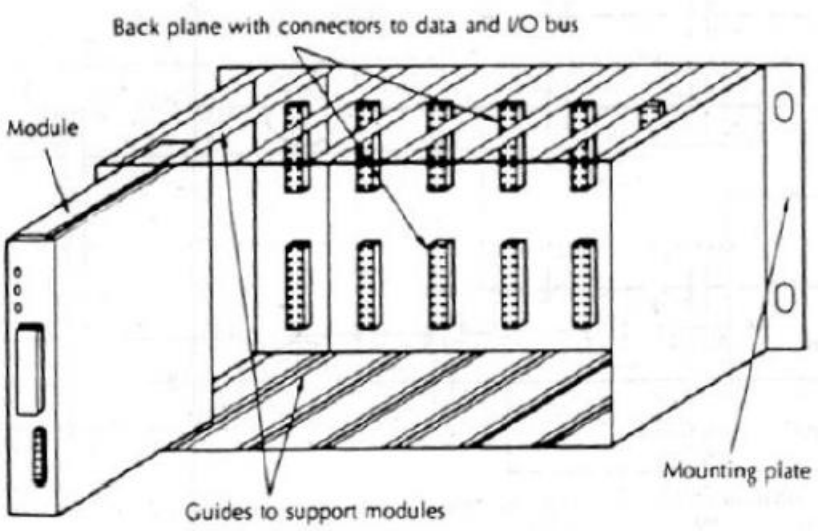
(б) модуларни ПЛЦ системи.

Компактни ПЛЦ контролери су независни, затворени уређаји са фиксним бројем улаза/излаза, без могућности проширења (Сл. 4-5). У једном кућишту, обично мањих димензија, смешетни су: извор напајања, процесорска јединица и улазни и излазни модул. Компактни ПЛЦ контролери представљају економично решење, предвиђено за управљање системима и процесима мале сложености. Типично, поседују до 16 улаза и 16 излаза и меморију од неколико КБ.



Слика 4. Компактни ПЛЦ.

Модуларни ПЛЦ системи се састоје од већег броја модула који су смештени унутар механичког оквира, тј. шасије, који се зове рек Рек поседује већи број слотова за смештање модула. Сваки слот чини пар вођица дуж горње и доње странице река које служе за механичко учвршћење модула као и конектор на задњој плочи река за прикључује модула на заједничку магистралу изведену на штампаној плочи задње странице река. По правилу, први слот је намењен модулу извора напајања, који се прикључује на мрежни напон (220Вац) и генерише једносмерне напоне потребне за рад остатка система. Следећи, други слот се користи за модул логичке јединице, тј. процесорски модул који извршава кориснички програм и управља радом осталих модула. Преостали слотови се користе за модуле специјалне намене, као што су У/И модули, меморијски модули и сл. Овакав начин контрукције омогућава лако проширење система. На пример, ако је потребно повећати број улаза/излаза довољно је уградити додатни У/И модул. Или, ако због повећаних захтева обраде, постојећи процесорски модул више није одговарајући, он се може заменити новим, моћнијим, а да при томе остали модули не морају бити замењени. Број слотова у једном ПЛЦ реку је, типично, од 4 до 16. Могућност проширења ПЛЦ система није ограничена само на један рек. Уз помоћ посебних модула за проширење могуће је повезати два или више река, што омогућује да се једним процесорским модулом управља великим бројем додатних модула.[6]



Слика 5. Модуларни ПЛЦ.

На слици 5 је приказан модуларни ПЛЦ, у пракси се најчешће користи, јер и сам ПЛЦ користимо јер има ту способност великог броја улазно излазних јединица.

5.8 CPU и PM

ЦПУ је „срце“ ПЛЦ система. На Сл. 4-7 је приказана типична ЦПУ јединица у споју са програматором/монитором (ПМ) облика ручног терминала. ЦПУ јединца може бити већа или мања од оне приказане на Сл. 4-7, зависно од величине и сложености процеса којим управља. Један од главних параметара о коме се мора водити рачуна приликом избора ЦПУ јединице јесте величина интерне меморије. За управљање

једноставним процесом, довољан је мали ПЛЦ са ограниченом меморијом; за управљање већим системом, неопходан је већи ПЛЦ, који подржава напредније функције и поседује већу количину меморије. Код неких ПЛЦ -ова постоји могућност да се накнадно угради додатна меморија; код других та могућност не постоји. На самој ЦПУ јединици обично се могу наћи различити конектори за каблове који служе за повезивање са другим ПЛЦ-овима. Многе ЦПУ јединице поседују уграђену тзв. бацкуп батерију, која омогућавају да се у случају нестанка екстерног напајања сачува учитани кориснички ледер програм. За разлику од оперативног систем ПЛЦ-а који чува се у перманентној меморији (РОМ), кориснички програм се чува у РАМ-у, који, као што знамо, губи уписани садржај оног момента када се напајање искључи. Када ПЛЦ детектује престанак екстерног напајања, бацкуп батерија се аутоматски укључује да би се обезбедио напон напајања за РАМ и тако сачувао његов тренутни садржај до доласка екстерног напајања.[6]

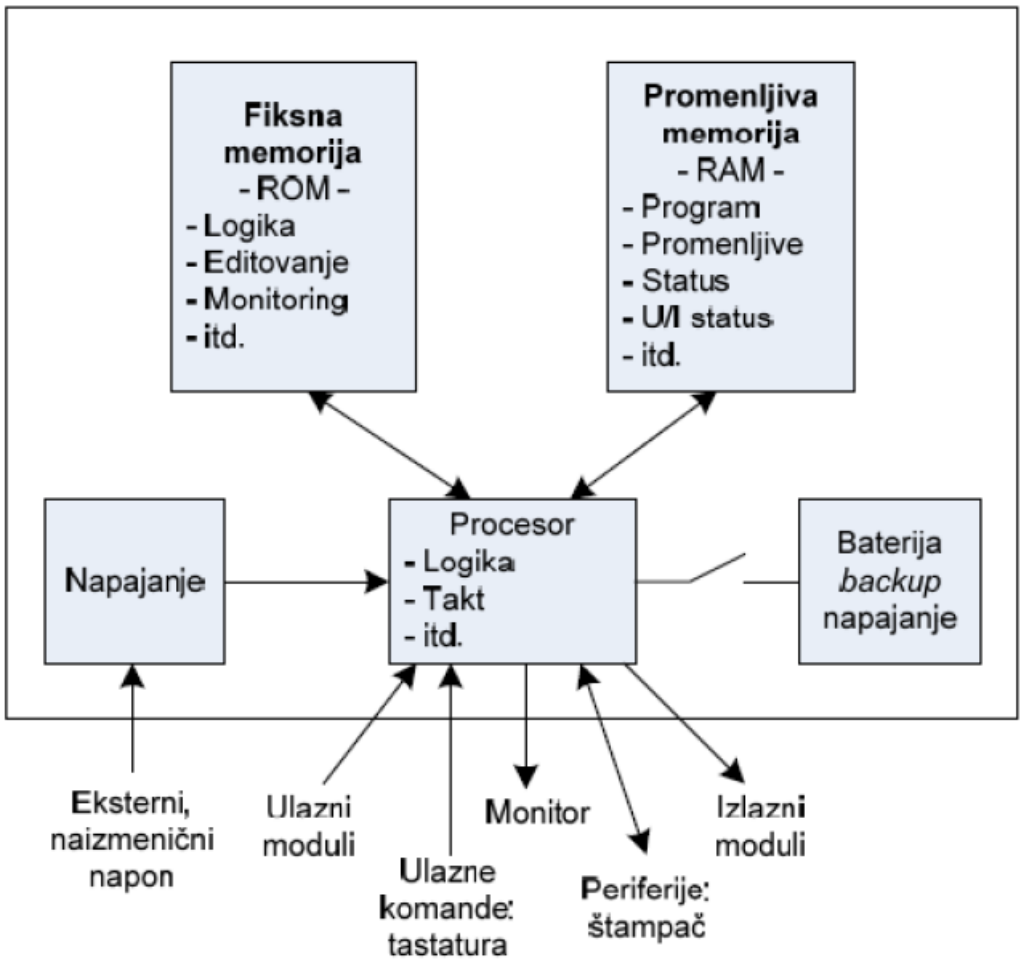
5.9 ПЛЦ улазни и излазни модули

ПЛЦ прибавља информације из окружења посредством улазних, а предаје информације окружењу путем излазних модула. Приступне тачке (терминали) улазног модула примају сигнале сензора и претварача. На приступним тачкама излазног модула генерише се напон за побуду актуатора (мотори, склопке, ...) и уређаја за индикацију (светиљке, дисплеји, ...). УИ модул може имати 4, 8, 12 или 16 терминала. Модул може бити улазни, излазни или комбиновани (У/И) са пођеднаким или различитим бројем улазних и излазних терминала (нпр. 12 улаза и 8 излаза). Код мањих система, ЦПУ, улазни и излазни модули се смештају у исти рек. Код већих ПЛЦ система, улазни и излазни модули су смештени у посебне рекове који су са ЦПУ-ом повезују помоћу одговарајућег вишежичног кабла. Најважнија карактеристика У/И модула је опсег напона или струје који модул прихвата, односно генерише. Напонски и струјни опсег модула мора бити усаглашен са електричним карактеристикама система или уређаја с којим се ПЛЦ модул повезује. Да би се задовољили најразличитији захтеви у погледу напонских и струјних опсега, произвођачи ПЛЦ система нуде палете улазних и излазних модула декларисаних за различите напонске/струјне опсеге. Поред дискретних улазних и излазних модула (прихватају и генеришу дискретне - дигиталне, тј. ОН/ОФФ сигнала), у употреби су и аналогни улазни и излазни модули који прихватају и генеришу аналогне сигнале. Такви модули поседују уграђене А/Д, односно Д/А конверторе.[7]

5.10 Структура ПЛЦ-а

Без обзира на величину ПЛЦ-а, централна процесорска јединица, тј. ЦПУ, или само процесорска јединица, увек обједињује микропроцесор и меморију. Код већих ПЛЦ-ова, ЦПУ садржи само микропроцесор и меморију, док код мањих, додатно, садржи У/И интерфејс и извор за напајање. Такође, могуће је да ЦПУ садржи микропроцесор, меморију и извор напајања, а да је У/И интерфејс реализован

екстерним У/И модулима. Једна таква организација, приказана је блок дијаграмом са Сл. 4-9. Фиксна меморија садржи програм (или програме) које је у поставио произвођач. Ово су програми оперативног система ПЛЦ-а, који имају сличну намену, мада са неупоредиво мањим могућностима, као нпр. ДОС оперативни систем код старијих ПЦ рачунара, Њиндоњс код новијих или УНИХ код радних станица. Програми оперативног система су смештени у посебном меморијском интегрисаном колу типа РОМ. Као што знамо, садржај РОМ-а се не може брисати нити мењати, а остаје непромењен и након искључења напајања. Променљива меморија, реализована помоћу РАМ меморијских чипова, подељена је на већи број секција које садрже кориснички програм, програмске променљиве, тренутне статусне информације контролера итд.



Слика 6. Структура ПЛЦ-а.

На слици 6. приказана је структура једног ПЛЦ уређаја, његове процесорске јединице, напајања, батерије као и своје меморије.[7]

5.11 Скен циклус

Оперативни систем ПЛЦ контролера је пројектован тачно за одређену врсту примене. Наиме, претпоставља се да ће у својој основној форми, ПЛЦ бити коришћен за реализацију извесних логичких функција које пресликавају сигнале са сензора у

сигнале који се преносе на актуаторе. Отуда се од ПЛЦ-а очекује да периодично очитава (уноси) сигнале са сензора, извршава одређен број аритметичко-логичких операција (у складу са задатом функцијом) чији резултати се преносе на извршне органе или неке друге индикаторске уређаје. Поред тога, са истом или неком другом учестаношћу, ПЛЦ треба да одржава комуникацију (размењује податке) са неким другим рачунарским системима у мрежи.

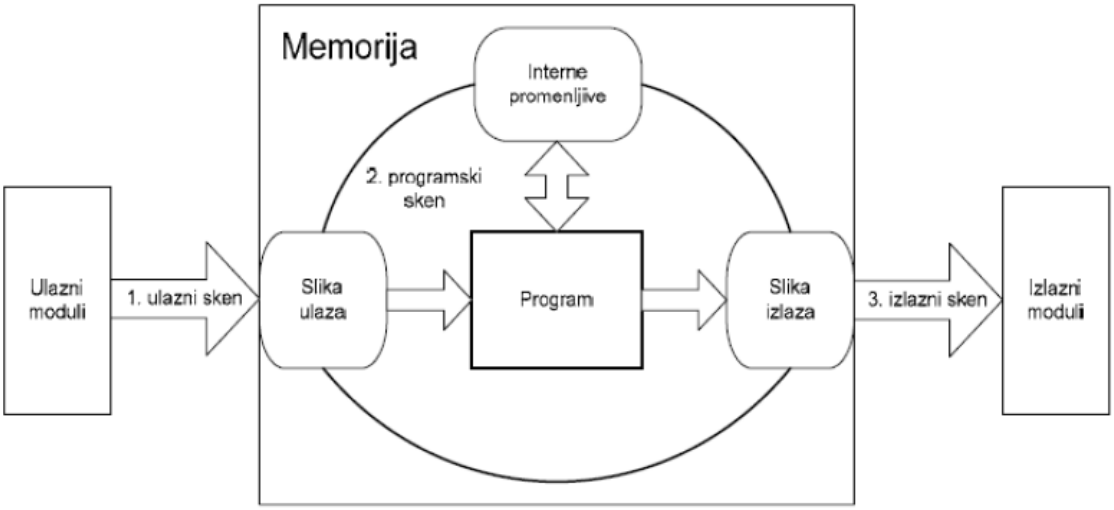
Полазећи од овог захтева, оперативни систем ПЛЦ контролера пројектован је тако да, у току рада система, аутоматски обезбеди циклично понављање наведених активности (Скен циклус) као што је то илустровано на Сл. 7.

Слика 7. Скен циклус ПЛЦ контролера.

Скен циклус започиње са улазним скеном у оквиру кога ПЛЦ очитава садржај улазних линија (регистара улазних модула). Очитани подаци се преносе у одређено подручје меморије – слика улаза. Затим се активира програмски скен у оквиру кога процесор извршава програмске наредбе којима су дефинисане одговарајуће аритметичко-логичке функције. Подаци (операнди) који се користе у програмским наредбама узимају се из меморије и то из подручја означеног као слика улаза (ако су операнди улазни подаци) или из подручја где се смештају интерне променљиве. Резултати обраде се смештају у посебно подручје меморије – слика излаза. Важно је истаћи да се при извршавању програмских наредби подаци не узимају директно са улазних модула, нити се резултати директно постављају на излазне модуле, већ

програм размењује податке искључиво са меморијом. По завршетку програмског скена, оперативни систем ПЛЦ контролера активира излазни скен у оквиру кога се подаци из слике излаза преносе на излазне линије (регистре излазних модула). На овај начин ствара се утисак да је ПЛЦ све операције дефинисане програмом обавио у исто време.

Четврти део скен циклуса – комуникација - намењен је реализацији размене података са уређајима који су повезани са ПЛЦ-ом. Након тога, оперативни систем доводи ПЛЦ у фазу одржавања у оквиру које се ажурирају интерни тајмери и регистри, обавља управљање меморијом као и низ других послова везаних за одржавање система, о којима корисник и не мора да буде информисан. У зависности од типа уграђеног микропроцесора улазни и излазни скен циклус извршавају се у времену реда милисекунди (од 0.25мс до 2.56ms). Трајање програмског скена, свакако зависи од величине програма.[6]



Слика 8. Размена података за време скен циклуса.

На слици 8. Видимо приказ размене података за време једног скен циклуса, као и пут скена од улаза ка излазу, тјст од улазне јединице до излазне јединице.

5.12 Карактеристике CPU јединице

Као што је већ речено, ЦПУ модул садржи микропроцесор и меморију. Микропроцесор обухвата аритметичко-логичку јединицу (АЛУ), регистре и управљачку јединицу. У функционалном смислу микропроцесор ПЛЦ-а се битно не разликује од микропроцесора било ког микрорачунара опште намене. Међутим, разлика постоји у програмском језику; док се за програмирање микрорачунара користи асемблерски језик или неки виши програмски језик (нпр. Ц), за програмирање ПЛЦ-ова се користи, као што је раније речено, језик ледер дијаграма. У основи, основна

разлика се огледа у скупу наредби који је одабран тако да се задовоље основни захтеви у погледу коришћења ПЛЦ-а.

Основне карактеристике процесорског модула изражавају се преко следећих елемената:

Меморија(РАМ) – карактерише се својом величином, могућношћу проширења и конфигурисања за смештања програма или података.

У/И тачке – карактерише се највећим бројем локалних У/И адреса које подржава процесор у току улазног и излазног скена, као и могућношћу проширења преко удаљених У/И. (Под удаљеним У/И подразумева се посебна шасија која садржи У/И модуле који размењују податке са ПЛЦом).

Комуникационе опције – односе се на разноврсност уређаја за спрегу (комуникационог интерфејса) који подржавају различите топологије мрежа и различите комуникационе протоколе.

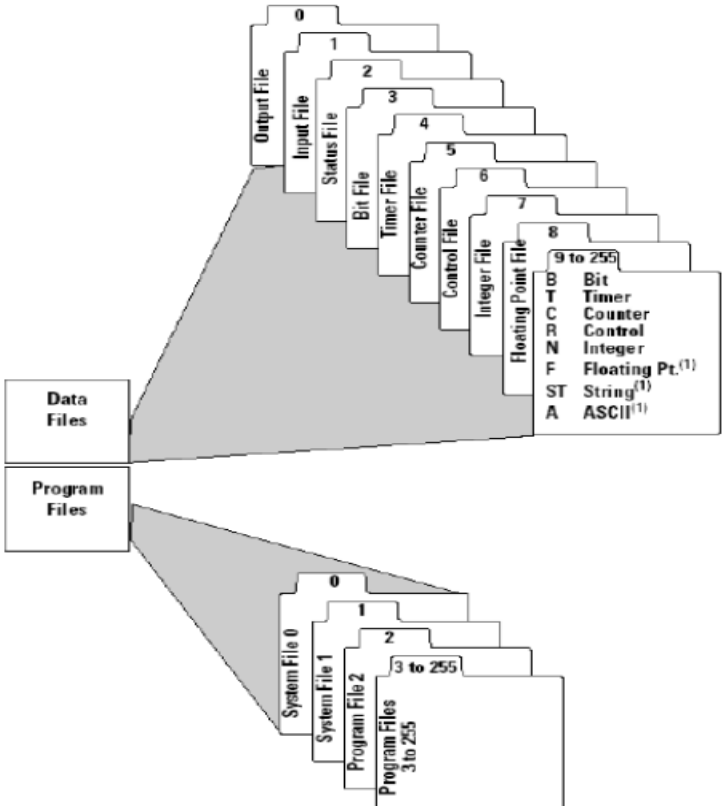
Опције трајног памћења – односе се на расположивост различитих типова меморијских ЕПРОМ модула који обезбеђују трајно памћење података.

Перформанса – специфицира се преко времена програмског скенирања потребног за извршење 1Кбајт програма, преко времена потребног за улазни и излазни скен, као и времена извршавања једне бит наредбе.

Програмирање – специфицира се у односу на број подржаних различитих наредби ледер језика.[6]

5.13 Оргнизација меморије

Оперативни систем ПЛЦ контролера, који реализује скен циклусе, управља и заузећем РАМ меморије, која је организована на посебан начин. У принципу, РАМ меморија се дели на програм филес (програмске датотеке) и дата филес (датотеке података) Сл. 4-12. Скуп програма и датотека података које су формиране за једну апликацију чини процессор филе (процесорску датотеку). Она садржи све наредбе, податке и спецификацију модула који су релевантни за дату апликацију, односно кориснички програм. Процесорска датотека чини једну целину која се може преносити са једног процесорског модула на други. То заправо значи да се једна апликација може развити на једном систему и затим у целини пренети и користити на другом систему.[5]



Слика 9. Организација меморије ПЛЦ контролера.

На слици 9. Приказана је ораганизација меморије ПЛЦ контролера, која се дели на програске датотеке и датотеке података.

5.14 Програмске датотеке

Програмске датотеке садрже информације о самом контролеру, главни кориснички програм и потпрограме. Свака апликација (процесорска датотека) мора да има следеће три програмске датотеке:

System Program – sistemski program (file 0)- садржи различите информације о самом систему као што су тип процесора, конфигурација У/И модула, име процесорске датотеке, лозинку и низ других релевантних података.

Reserved– датотека резервисна за потребе оперативног система (филе 1)

Main Ladder Program – glavni leder program (file 2)– главни ледер програм (филе 2) – програм који формира сам корисник и у оквиру кога се дефинише низ операција које ПЛЦ треба да изведе.

Subroutine Ladder Program - potprogrami (file 3 - 255)– кориснички потпрограми који се активирају у складу са наредбама за њихово позивање које се налазе у главном програму.

Датотеке података садрже податке који се обрађују помоћу наредби ледер програма. При томе се под појмом подаци подразумевају конвертоване (нумеричке) вредности сигнала који се преко улазно/излазних модула уносе у контролер, или се из контролера преносе на излазне уређаје, као и интерне променљиве које се користе као операнди у различитим операцијама.

Датотеке података организоване су у складу са типом променљивих које садрже. То заправо значи да једна датотека садржи само један тип (врсту) података. Једна процесорска датотека може да има највише 256 датотека података.

5.15 Типови променљивих и датотека

Основна карактеристика датотеке података је њен тип. Као што је већ истакнуто тип датотеке, заправо указује на врсту променљивих које се у њој памте. То надаље подразумева да тип датотеке уједно одређују и њену организацију, која зависи од врсте податка и усвојеног начина за његово приказивање у рачунару.

Датотека се означава помоћу редног броја, који једнозначно одређује место те датотеке у низу датотека података које се налазе у процесорској датотеци и слова којим се идентификује тип датотеке. Првих девет датотека имају унапред дефинисан тип који не може да се мења. Типове преосталих датотеке корисник сам одабира и дефинише у складу са апликацијом коју развија.

File 0 – Tip O - output (излаз) – садржи слику излаза; садржај датотеке се преноси на излазне линије за време излазног скена.

File 1 – Tip I - input (улаз) – садржи слику улаза; у ову датотеку се за време улазног скена смештају вредности са улазних линија.

File 2 – Tip S - status– садржи податке везане за рад контролера.

File 3 – Tip B - bit– садржи интерне променљиве бит типа.

File 4 – Tip T - timer (часовник) – садржи податке који се користе за интерне часовнике.

File 5 – Tip C - counter (бројач) – садржи податке који се користе за интерне бројаче.

File 6 – Tip R - control (управљање) – садржи дужину, положај показивача и битове статуса за одређене наредбе као што су наредбе за померање садржаја регистара и секвенци.

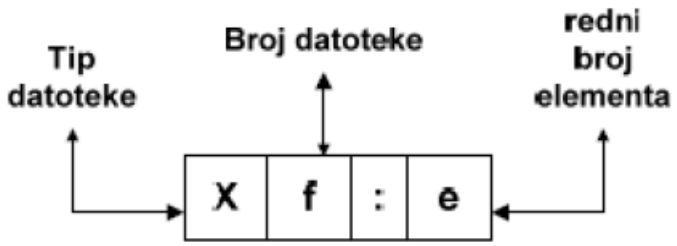
File 7 – Tip N - integer (целобројна) – садржи податке целобројног типа.

File 8 – Tip F - floating point (реална) – садржи податке представљене у формату покретног зареза као 32-бит бројеве у опсегу (1.1754944e-38 то 3.40282347e+38).

File 9 do file 255 – Tip дефинише корисник - корисничке датотеке – ове датотеке дефинише корисник као датотеке типа B, T, C, N. [6]

5.16 Елемент датотеке

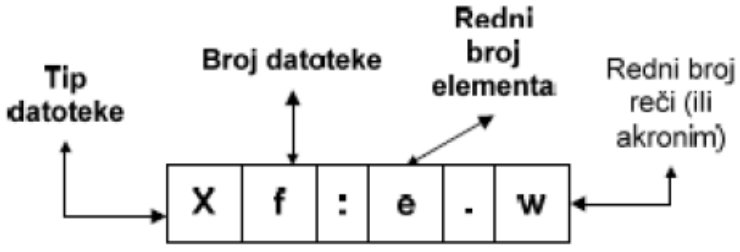
Основна јединица датотеке је један елемент. Сваки елеменат се састоји из неколико (једне или више) 16-битних речи. Број речи које чине један елемент зависи од типа датотеке, односно врсте података који се у њу смештају. Као што је већ истакнуто, подаци који су смештени у датотекама представљају операнде (променљиве) који се користе у појединим програмским наредбама. Ове променљиве се у програму позивају преко својих симболичких имена која представљају логичке адресе. При томе, адресе омогућавају да се позове не само елеменат у целини, већ и његов део. То значи да се могу адресирати поједине речи у оквиру елемента или поједини битови у оквиру речи. Будући да су подаци у извесном смислу хијерархијски организовани: 1 елеменат садржи неколико речи, а 1 реч 16 битова, то су и одговарајуће адресе структуиране по истом хијерахијском принципу. Поједине речи и битови у неким датотекама имају и придружене акрониме (словне скраћенице), што додатно олакшава њихово коришћење. Адреса елемента – у принципу, сваки елемент у оквиру датотеке се идентификује помоћу његовог релативног положаја у односу на почетак датотеке (нулти, први, други, … елемент). У складу с тим адреса елемента има изглед као на Сл. 10.



Слика 10. Адреса елемента.

На слици 10. Приказана је структура адресе елемента, која се састоји из типа датотеке, броја датотеке као и редног броја елемента.

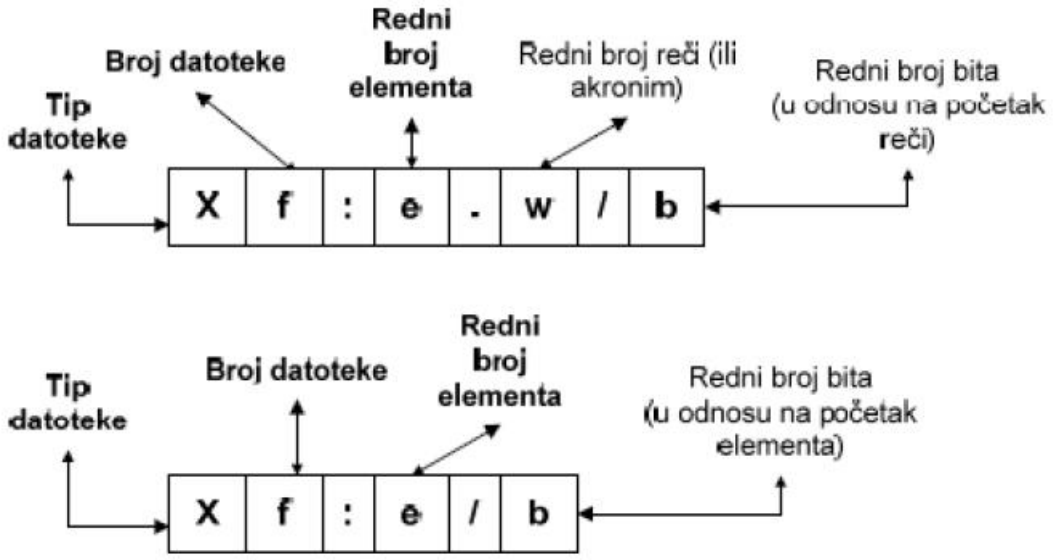
Адреса речи – једна реч елемента се идентификује или помоћу релативног положаја те речи у оквиру елемента, или помоћу посебног акронима (уколико је исти дефинисан). Формат адресе једне речи приказан је на Сл. 11.



Слика 11. Адреса речи.

Адреса бита – Један бит у оквиру речи идентификује се или преко његовог релативног положаја у оквиру те речи (нулти, први, други, … бит бројано с десна у

лево) или преко релативног положаја у односу на почетак одговарајућег елемента коме припада реч чији се бит адресира (Сл. 12).



Слика 12. Адреса бита.

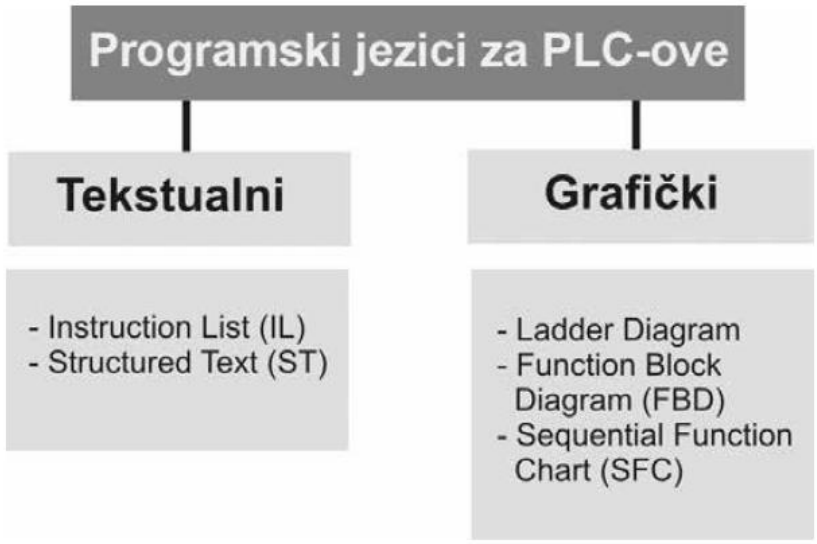
На слици 12. Приказана је структура бита и разлика у структури адресе бита и адресе речи.

5.17 Програмски језици за ПЛЦ-ове

Старији ПЛЦ-ови су се углавном програмирали помоћу лествичастих дијаграма који су по својој структури подсећали на шематске дијаграме којима су се објашњавале релејне мреже. Модерни ПЛЦ-ови су донели велике новине и побољшања, како у својој хардверској структури, тако и у софтверским решењима. Данашњи ПЛЦ-ови се програмирају на више начина па је зато била потребна и стандардизација програмских језика. У складу са захтевима које су наметнули тржиште, корисници, па и сами произвођачи ПЛЦ-ова, усвојен је интернационални стандард ИЕЦ 61131-3 којим је дефинисано и дозвољено следећих пет програмских језика за програмабилне логичке контролере:

1. Instruction List (IL)– листа наредби
2. Structured Text (ST)– структуриран текст
3. Ladder Diagram– лествичасти дијаграм
4. Function Block Diagram (FBD)– функцијски блок дијаграм
5. Sequential Function Chart (SFC)– секвенцијални функцијски графикони.

Прва два језика с ове листе припадају класи текстуалних језика, при чему је ИЛ програмски језик у класи асемблерских језика, док је СТ програмски језик сличан Пасцал програмирању. Остала три језика припадају класи графичких језика, међу којима је језик лествичастих дијаграма (Ладдер Диаграм) и даље убедљиво најзаступљенији програмски језик када је у питању програмирање ПЛЦ-а (13). [7]



Слика 13. Програмски језици за ПЛЦ-ове.

На слици 13. Видимо приказ и поделу програмских језика који се могу користити у програмирању једног ПЛЦ уређаја.

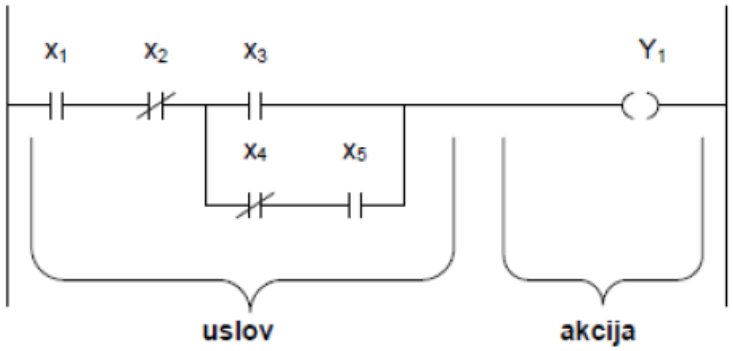
5.18 Ледер програмирање

Ако се ПЛЦ посматра као микропроцесорски систем, што он и јесте, онда би се могло очекивати да се за његово програмирање користе стандардни програмски језици. Међутим, ако се пође од чињенице да је ПЛЦ пројектован као наменски микропроцесорски систем за управљање и надзор рада неког процеса, и да у складу с тим има посебан оперативни систем који обезбеђује периодично понављање скен циклуса, онда је логично очекивати да је за његово програмирање развијен и посебан програмски језик. Као што је већ раније истакнуто, ПЛЦ је почетно развијен са идејом да замени релејне системе. То значи да се очекивало да он реализује одговарајућу временску секвенцу логичких операција. Поред тога, успешна примена ПЛЦ-а у пракси, захтевала је и да се његово програмирање прилагоди техници која је свим корисницима релејних система добро позната. Из свих ових разлога, за пројектовање ПЛЦ-ова

развијен је програмски језик заснован на ледер (лествичастим) дијаграмима – ледер програмски језик.

5.19 Ранг

Једна програмска линија ледер језика састоји се из низа графичких симбола (програмских наредби) који представљају различите логичке елементе и друге компоненте као што су тајмери и бројачи, који су поређани дуж хоризонталне линије – ранг (рунг) – која је на оба краја спојена са двема вертикалним линијама. Према томе, ледер дијаграм има изглед лествица, одакле потиче и његов назив (ладдер – лествице). Сваки ранг ледер дијаграма састоји се из два дела. На левој страни ранга налази се услов изражен у форми контактне (прекидачке) логике, док се на десној страни ранга налази акција која треба да се изврши уколико је услов испуњен (труе) (Сл. 4-16). Услов – Графички симболи на левој страни ранга односе се или на стања сигнала који представљају физичке улазе ПЛЦ-а, и чије су вредности током улазног дела скен циклуса смештене у слици улаза (инпут имаге филе), или на стања интерних променљивих, чије су вредности смештене у одговарајућим датотекама. Сваки симбол представља једну унарну бинарну операцију којој је придружена одговарајућа таблица истинитости. Уз графички симбол назначава се и адреса променљиве која представља операнд. При испитивању истинитости услова сматра се да се над свим симболима у једној линији (редна, серијска веза) обавља логичка "И" операција. То значи да је услов истинит уколико је сваки појединачни исказ истинит. На левој страни ранга дозвољена су и грањања (паралелене везе). При испитивању истинитости услова паралелене везе се третирају као логичка "ИЛИ" операција. То значи да ће исказ представљен низом паралелних грана бити истини, ако бар једна од грана садржи истинит исказ. Потребно је да се истакне да лева страна ранга може бити формирана и тако да на њој нема ни једног симбола. У том случају сматра се да је услов који се на тај начин дефинише увек истинит.



Слика 14. Ледер ранг.

Акција – Графички симболи на десној страни ранга односе се или на физички излаз (променљиве смештене у слици излаза (оутпут имаге филе), које ће бити пренете на излазе контролера у току излазног дела скен циклуса) или на интерне променљиве, чије су вредности смештене у одговарајућим датотекама. Сваки симбол представља једну наредбу која се извршава ако је услов на левој страни истинит. Уз симбол се назначава и адреса променљиве чија се вредност мења приликом извршавања наредбе,

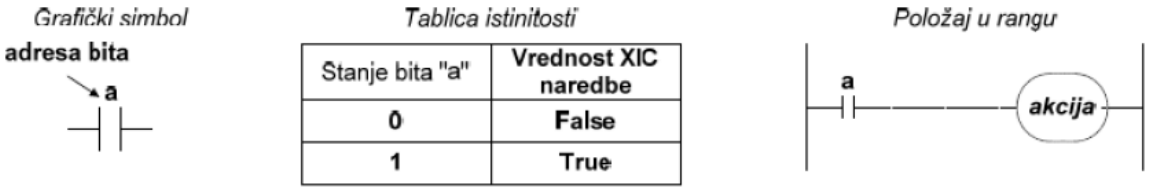
или која на било који други начин учествује у реализацији наредбе (нпр. отпочињање или заустављање неке активности, скок на неки други ранг, позив потпрограма итд.). Серијска веза на десној страни ранга није дозвољена, док паралелна веза означава да се више различитих наредби извршавају као резултат испитивња истинитости једног истог услова.

У литератури је уобичајено да се и симболи који означавају услов и симболи који означавају акцију означавају као наредбе. Отуда је неопходно да се истакне суштинска разлика између наредби услова и наредби акције. Наиме, извршавање наредби услова обавља се тако што се у зависности од вредности операнда, према придруженој таблици истинитости, наредби додељује вредност (0 или 1). Дакле, наредбе услова се извршавају у сваком скен циклуса и резултат њиховог извођења је вредност наредбе. За разлику од тога наредбама акције се или додељује вредност некој променљивој или извршава нека друга активност. Ове наредбе се извршавају само ако је услов који им претходи истинит (додељена му је вредност 1). При томе се самим наредбама акције не додељује никаква вредност.

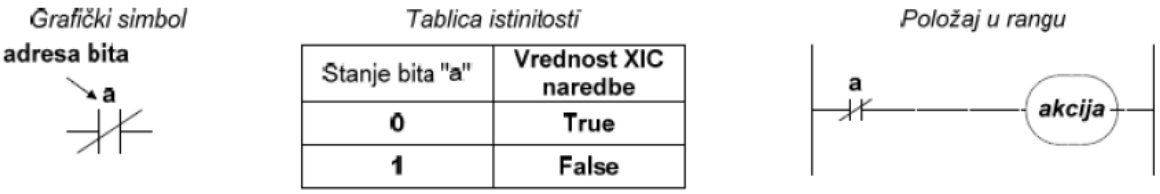
Ледер програм се извршава у току програмског дела скен циклуса и то тако што се обрађује ранг по ранг по редоследу како су они поређани у ледер дијаграму. У сваком рангу испитује се истинитост услова и уколико је услов истинит извршавају се одговарајуће наредбе у десном делу ранга. То значи да променљиве у десном делу ранга могу мењати своју вредност само једанпут у току скен циклуса, и то управо онда када се одговарајући ранг испитује. Потребно је запазити да уколико се променљива на десној страни ранга односи на физички излаз, вредност излаза неће бити промењена у истом тренутку времена. Наиме, за време програмског скена мењају се само вредности променљивих смештених у слици излаза. Тек касније, за време излазног дела скен циклуса, све променљиве из слике излаза биће пренете на одговарајуће излазне линије. Иста ствар важи и за улазне променљиве. Другим речима, за време програмског скена испитивање истинитости услова односи се на вредности променљивих у слици улаза, које су ту уписане за време улазног дела скен циклуса који је претходио програмском скену, а не на тренутне вредности променљивих на улазним линијама. Наравно, сви услови и наредбе који су везани за интерне променљиве извршавају се у тренутку сканирања појединог ранга.[5]

5.20 Бит наредбе

Бит наредбе су, као што само име каже наредбе чији су операнди битови. С гледишта локације операнада, то значи да се они најчешће налазе у датотеци 3 (бит филе - Б), дигиталним улазним или излазним датотекама (инпут имаге филе 1 или оутпут имаге филе 0) или у корисничким датотекама бит типа. Поред тога, адресирани операнд може да се налази и у било којој другој датотеци у оквиру које је могуће адресирати поједини бит. За време програмског скена у оквиру бит наредби испитује се стање појединог бита, или се његова вредност поставља на 1 (сет) или на 0 (ресет). Бит наредбе за дефинисање услова се постављају на левој страни ранга и дефинишу услов који се односи на стање бита чија је адреса дефинисана у наредби. Као резултат извођења наредба добија истиносну вредност труе (истинит) или фалсе (неистинит).



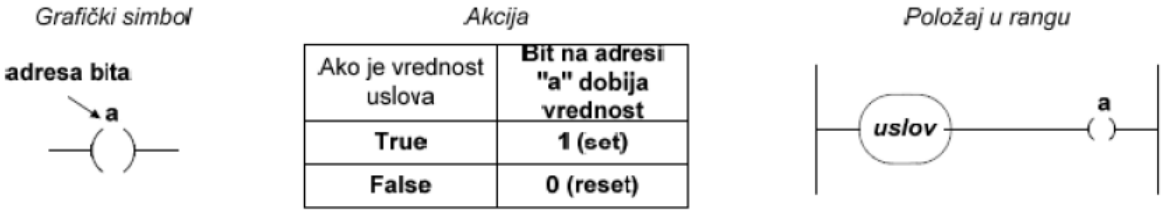
XIO - Examine if open (ispitivanje da li je kontakt otvoren)



Слика 15. Испитивање контакта.

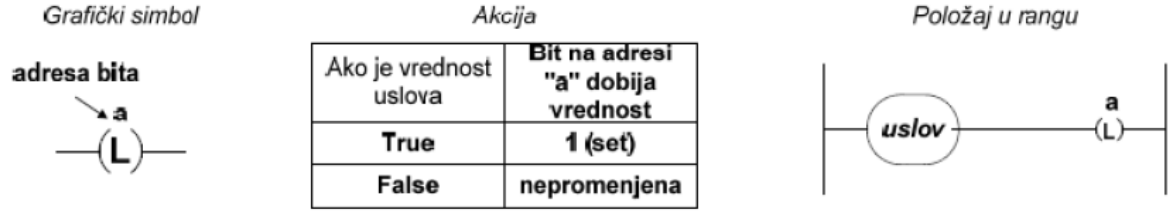
Називи ове две наредбе потичу од испитивања бинарних сигнала који долазе са прекидачких кола. У том смислу, XIC наредба се односи на нормално отворен прекидач, док се XIO наредба односи на нормално затворен прекидач.

Помоћу бита наредбе за постављање вредности излаза биту чија је адреса наведена у наредби додељује вредност 1 или 0. Подсетимо се да се ове нардбе налазе на десној страни ранга, што значи да ће се оне извршити само ако је исказ (услов) на левој страни ранга истинит.



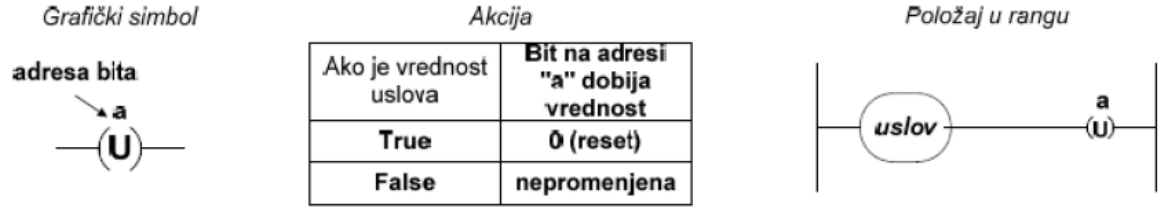
Слика 16. Побуђивање излаза

Потребно је да се запази да се овом наредбом вредност бита чија је адреса “a” може променити само једанпут за време скен циклуса. Ова вредност остаће неизмењена све до следећег скен циклуса, када ће се при скенирању одговарајућег ранга поново испитати услов и извести одговарајућа акција.



Слика 17, Памћење излаза.

OTL - Output latch или памћење излаза OTL наредбом се адресирани бит може искључиво поставити на 1. Наиме, за разлику од OTE наредбе којом се вредност бита може постављати на 0 или 1 сваки пут кад се ранг скенира, код OTL наредбе вредност бита се поставља (лечује) на 1 у првом скену у коме је услов истинит. Након тога ова наредба постаје неосетљива на истиносну вредност услова. То значи да ће вредност бита остати неизмењена без обзира на то како се мења вредност услова.



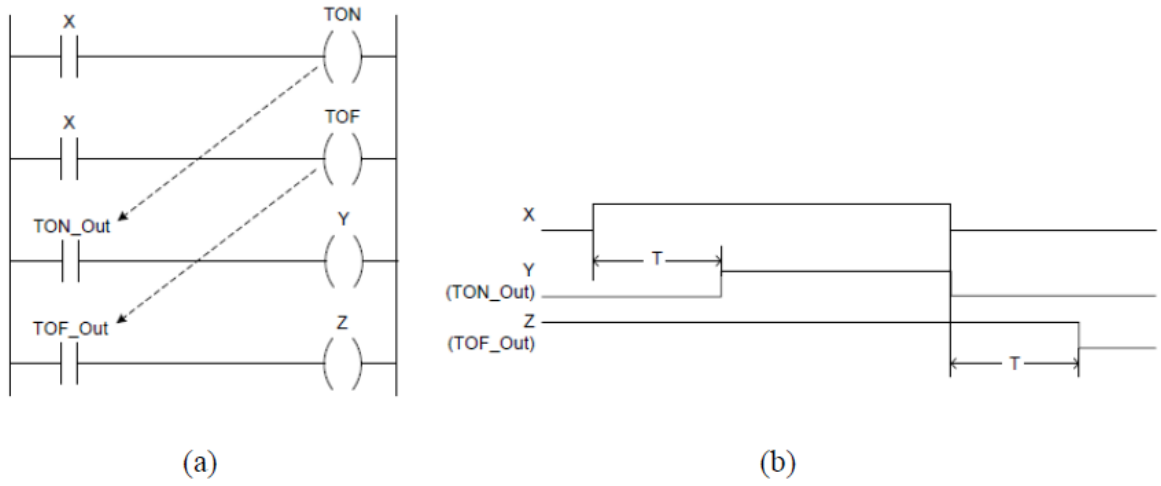
Слика 18. Ресетовање излаза.

OTU - Output unlatch или ресетовање излаза OTU наредбом се адресирани бит може искљу чиво поставити на 0. При томе, вредност бита се поставља (лечује) на 0 у првом скену у коме је услов испуњен. Након тога ова наредба постаје неосетљива на вредност услова. Потребно је да се истакне да се OTL и OTU наредба користе увек у пару, при чему се у обе наредбе адресира исти бит. [5]

5.21 Тајмери и бројачи

Приликом управљања или нацора процеса често је потребно да се нека активност отпочне или заустави после одређеног временског периода, или да се понови одређени број пута. У том смислу неопходно је да ПЛЦ контролер који ће се користити за управљање процесом пружи могућност за мерење времена и пребројавање догађаја. Пребројавање догађаја обавља бројач (цоунтер), који након регистровања унапред заданог броја догађаја генерише одговарајући сигнал. Мерење времена остварује се помоћу часовника, тј. тајмера (тимер). У суштини тајмер изражава време као умножак одређеног основног интервала (временска база) . То заправо значи да тајмер ради као бројач протока основних интервала и да након истека одређеног, унапред заданог интервала времена, генерише одговарајући сигнал. ПЛЦ наредбе за мерење времена су, после бит наредби, најчешће коришћене наредбе у ледер програмирању. Подршка за програмско мерење времена, у виду специјализованих, тзв.

тајмерских наредби, постоји код свих данас распложивих фамилија ПЛЦ контролера. Иако начин рада и могућности ових наредби зависе од типа ПЛЦ контролера, уопштено говорећи, у ледер програмирању се користе три врсте тајмера: делау_он тајмер (или ТОН), делау_офф тајмер (или ТОФ) и РТО тајмер. Тајмери су наредбе акција. Услов који претходи тајмеру управља тајмером тако што му омогућава или брани рад. Тајмер, за време док је омогућен, одбројава основне временске интервале и, након достизања задате вредности, поставља одређени бит (или битове) из интерне меморије ПЛЦ контролера, који се могу користити у другим ранговима за формирање услова. Сл. 4-18 илуструје разлику између делау-он и делау-офф тајмера. ТОН је омогућен док је услов тачан, а ТОФ док је услов нетачан. ТОН се искључује тренутно, а укључује по истеку задатог времена, Т. Насупрот томе, ТОФ се укључује тренутно, а искључује по истеку задатог времена, Т. (Овде се под укључивањем/искључивањем сматра сетовање/ресетовање бита који тајмер контролише – ТОН_ Оут, односно ТОФ_Оут). Другим речима, ТОН касни укључивање, а ТОФ искључивање. На пример, ТОН се може искористити да одложи по четак неког производног процеса за време потребно да се комора загреје; ТОФ се може искористити да вентилатор, који хлади комору, остане укључен неко задато време након што је систем искључен.



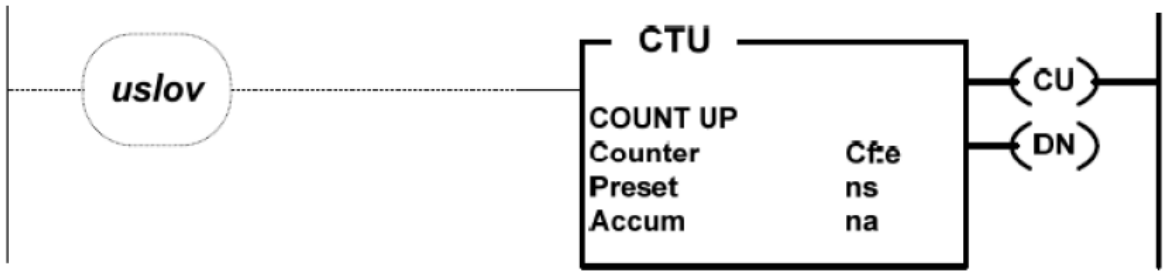
Слика 19. Delay-on и delay-off тајмер,(а) Ледер дијаграм; (б) Временски дијаграм.

TON и TOF тајмери увек почињу свој рад ″од нуле″, тј. промена услова на вредност која брани рад тајмера уједно и ресетује тајмер тако да када следећи пут тајмер буде пуштен у рад, његова почетна вредност биће 0. За разлику од тога, РТО тајмер (или, ретенциони тајмер) сумира (акумулира) време за које је омогућен. Када је поново омогућен, РТО наставља одбројавање основних временских интервала почев од вредности коју је имао у тренутку када је заустављен. За ресетовање РТО тајмера користи се посебна наредба.

Постоје два основна типа бројача бројач унапред (CTU – count up) и бројач уназад (CTD – count down). Обе наредбе су наредбе акције, што значи да се смештају у десни део ранга. Оба бројача броје промене вредности услова са неистинит на иситинит (узлазна ивица). При свим осталим вредностима услова, они задржавају пребројани износ и чекају следећи прелаз. Другим речима, бројачи се нити пуштају у рад, нити заустављају. Они непрекидно раде и бележе (броје) сваки прелаз истинит/неистинит. Достизање задате вредности се сигнализира постављавањем

одговарајућег бита – доне бит (DN) – на 1, али се бројање и даље наставља. Пребројани износ се може избрисати једино посебном РЕС наредбом.

Једина разлика између два типа бројача састоји се у томе што први (CTU) броји унапред од 0 до 32767, и поставља overflow бит (ОВ) на 1 кад пређе 32767, док други (CTD) броји уназад, од 0 до –32768, и поставља underflow бит (DN) кад пређе –32768.



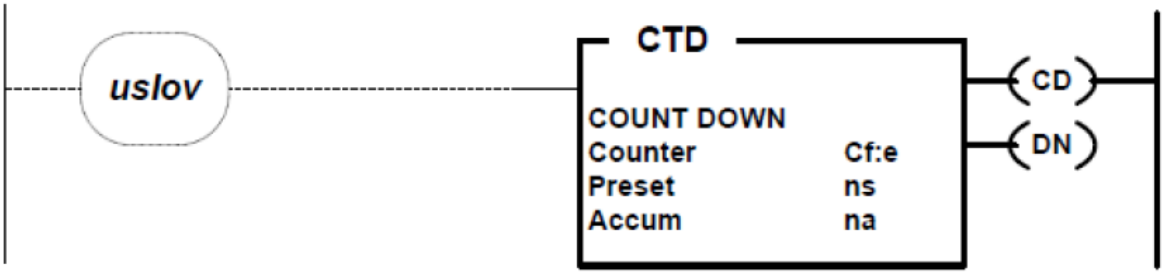
Слика 20. CTU, Count up.

Битови стања бројача мењају се у току програмског скен циклуса на следећи начин:

ОВ - Count up overflow бит се поставља на 1 када акумулирана вредност (ACC) прелази са 32767 на –32768 (у бинарној аритметици другог комплемента са 16-битном речи важи: 32767+1= – 32768), и наставља бројање унапреди.

DN - доне бит се поставља на 1 када је ACC≥PRE;

CU - count up enable бит се поставља на 1 када је услов истинит, а ресетује на 0 када је услов неистинит или када се активира одговарајућа RES наредба.



Слика 21. CTD, Count down.

Битови стања бројача мењају се у току програмског скен циклуса на следећи начин:

UN - Count down underflow бит се поставља на један када акумулирана вредност (ACC) прелази са – 32768 на 32767 (у бинарној аритметици другог комплемента, са 16-битном речи, важи: –32768-1= 32767), и наставља да броји уназад од те вредности.

DN - done бит се поставља на 1 када је ACC≤PRE;

CD - Count down enable бит се поставља на 1 када је услов истинит, а ресетује на 0 када је услов неистинит или када се активира одговарајућа RES наредба.

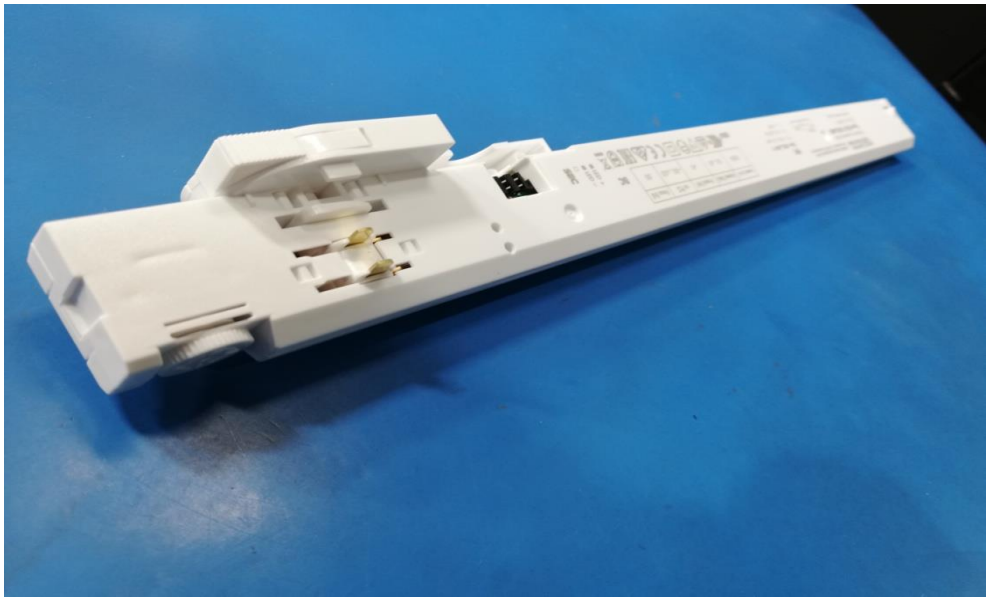
6. Опис рада

6.1 Разлог примене ПЛЦ уређаја

Долазимо до сржи овог дипломског рада а то је примена ПЛЦ контролисаних уређаја, као и употреба информационих система и база података у савременој индрустрији. Ту примену ћемо објаснити на пројекту израде Тестер машине, која се реализовала од јануара до јула месеца 2020 године. У компанији Vossloh Schwabe која се бави израдом решења везана за осветљење у погону и самој фабрици у Свилајнцу, Србији долази до потребе прављења нове тестер машине која ће се користити за тестирање драјвера који пали ЛЕД диоду и који као такав са све диодом може да се шета по ЛЕД шини, односно шинама на којима су постављене ЛЕД диоде и само осветљење. Разлог примене ПЛЦ уређаја је баш тај, да он у склопу машине која врши тестирања управља процесима у току тестирања и одређене податке складишти како би даље те податке искористили.

Слика 22. Компанија Vossloh Schwabe.[9]

Слика 23. Шина, in-track и осветљење.[10]



Слика 24. Производ који се тестира.

На фотографији 24. Приказан је производ, In-track, који се тестира и који служи да се на њега прикључи ЛЕД диода.

6.2 Тестер машина

Тестер машина се састоји из неколико делова. Први међу једнаким јесте Schleich, који служи да генерише високи напон међу спојевима и самим тим најбитнији тест који производ пролази јесте онај који генерише Шлајх који се састоји из великог броја релеја.

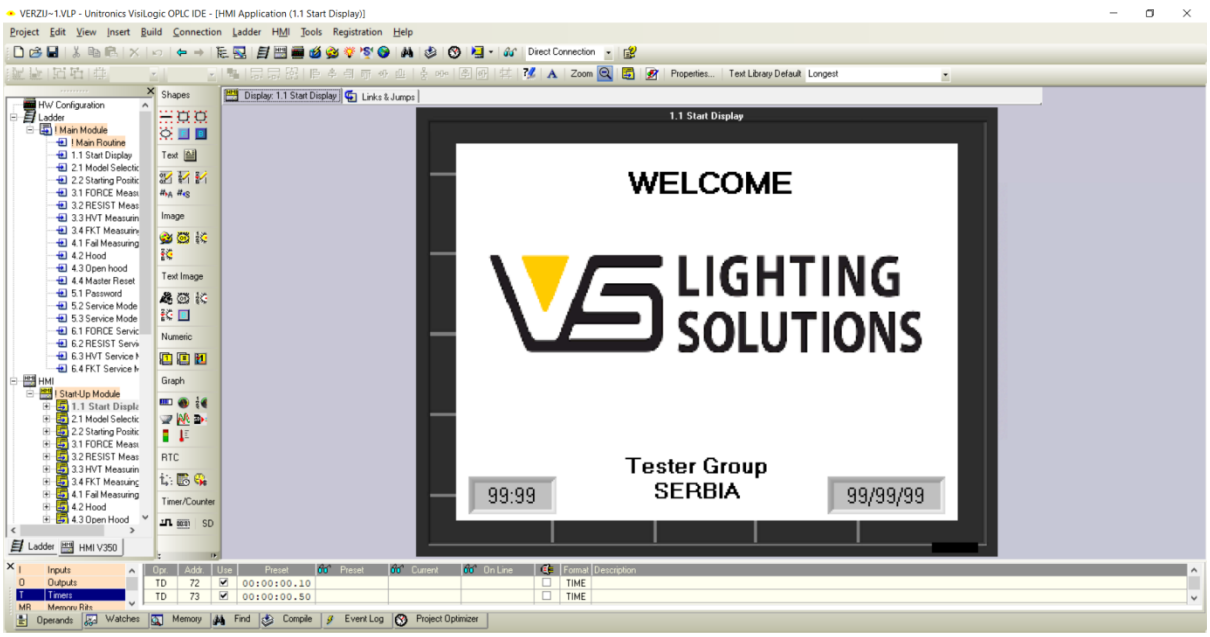
Слика 25. Тестер машина.

Слика 26. Шлајх.

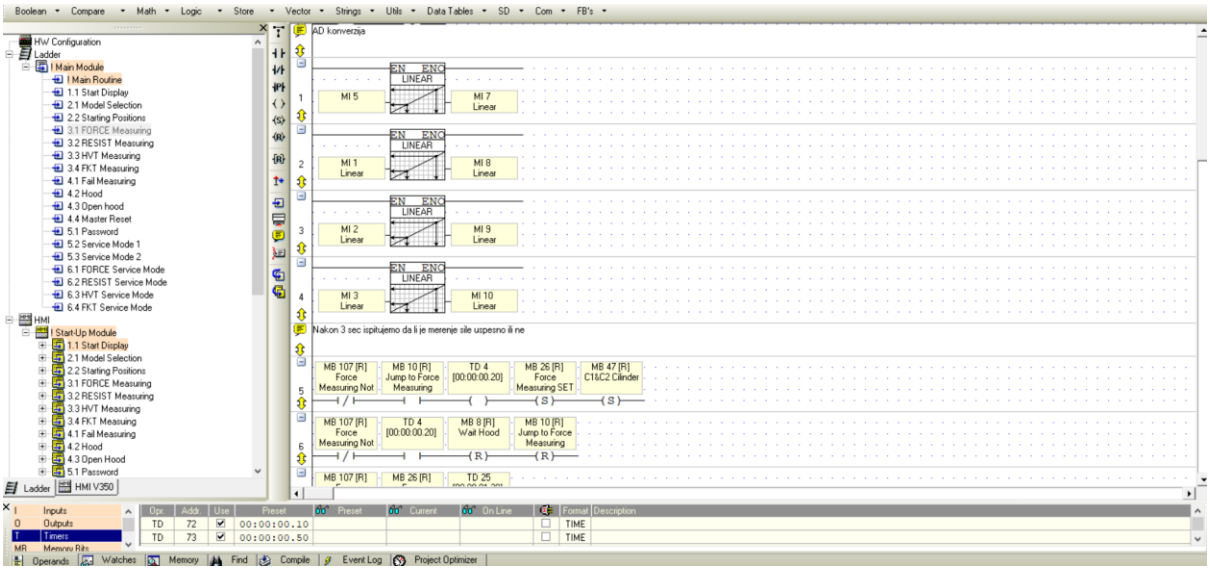
Након Шлајха који је везан за саму тестер машину, уређај који управља процесима јесте ПЛЦ. У овом случају коришћен је ПЛЦ фимре Unitronics и то модел v350. За рад са овим моделом користимо развојно окружење Visilogic. У нашем примеру пројекта, ПЛЦ који користимо је модуларног типа, где су његови модули смештени у кућиште тако да оператер који управља машином види само његов дисплеј.

Слика 27. ПЛЦ и његово кућиште.

ПЛЦ се програмира и моделује у програму Visilogic, и састоји се од сабрутина и приказа на дисплеју. Сабрутине можемо посматрати и као подпрограме који се могу позвати у одређеном делу ЛЕДЕР дијаграма.



Слика 28. Visilogic.



Слика 29. Ледер дијаграми.

На сликама 28,29. Приказан је програм који служи да се у њему програмира ПЛЦ уређај. Видимо два приказа и два приступа програмирања. Можемо на то посматрати као на Front end/Back end програмирање. Јер са једне стране имамо програмирање помоћу ледер дијаграма, док на слици 28. Видимо приказ и моделовање графичког окружења.

6.3 Процес тестирања

Тестирање уређаја обавља се у 5 пет етапа, и то:

- Тестирање опруга.

- Тестирање повезаности контаката са PBC.

- ФКТ1 тјст. Текст функционалности.

- ХВТ тјст. Тест високог напона.

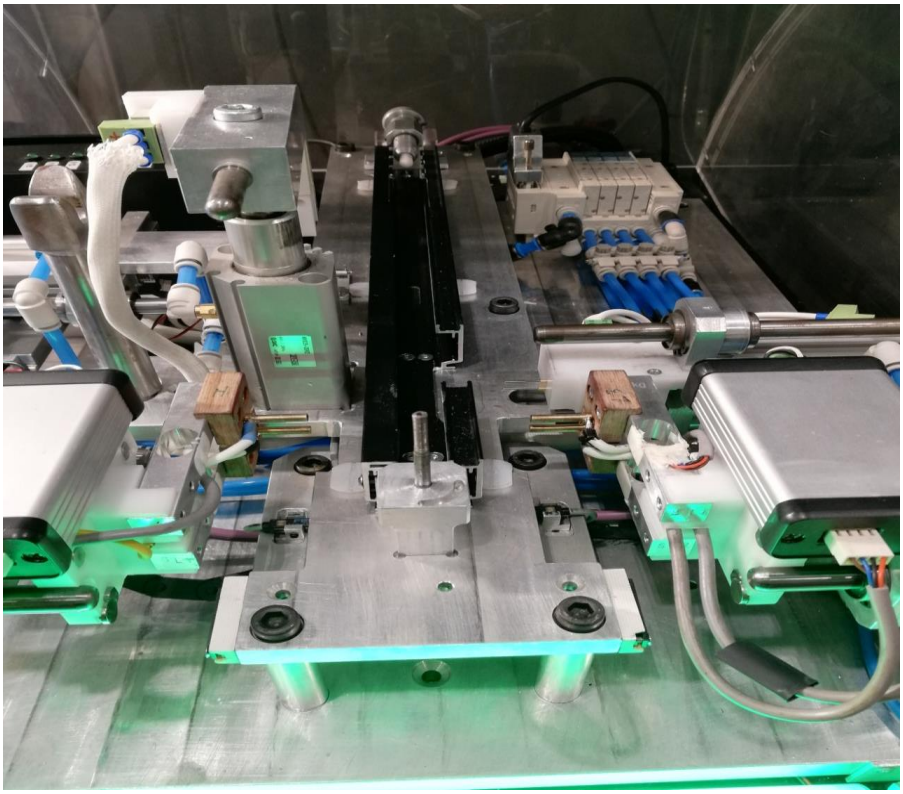
- ФКТ2 који представља други тест функционалности.

Да би се тестирање обавило успешно, оно што је потребно јесте да буду задовољени неки услови који су постављени. Испуњење тих услова видљиво је на дисплеју ПЛЦ уређаја. Оно што је битно јесте да радник приликом рада на машини, ништа сем одабира модела и постављања модела у положај за тестирање не ради. Што значи да сама тестирања и прорачуне обавља машина, и приказује раднику на дисплеју помоћу

ПЛЦ а одређене податке путем мреже складишти на серверу која се даље користи у процесу праћења и контроле производње и тестирања.

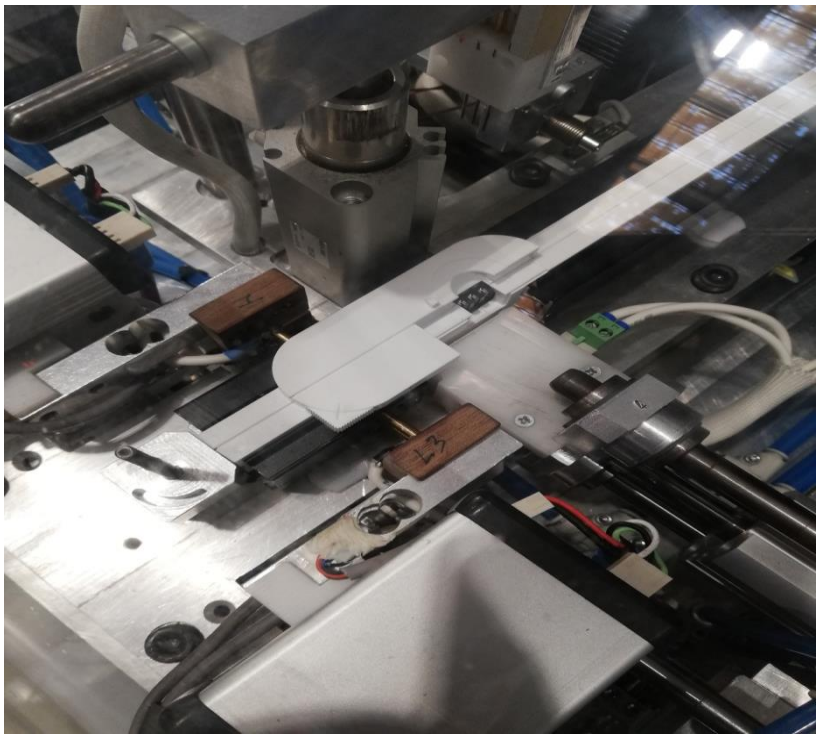
Слика 30. Почетни услови.

Процес тестирања протиче тако што радник поставља производ у почетни положај и затвара кутију, самим тим обезбеђујући себе и друге, јер се током тестирања постижу високи напонски нивои.

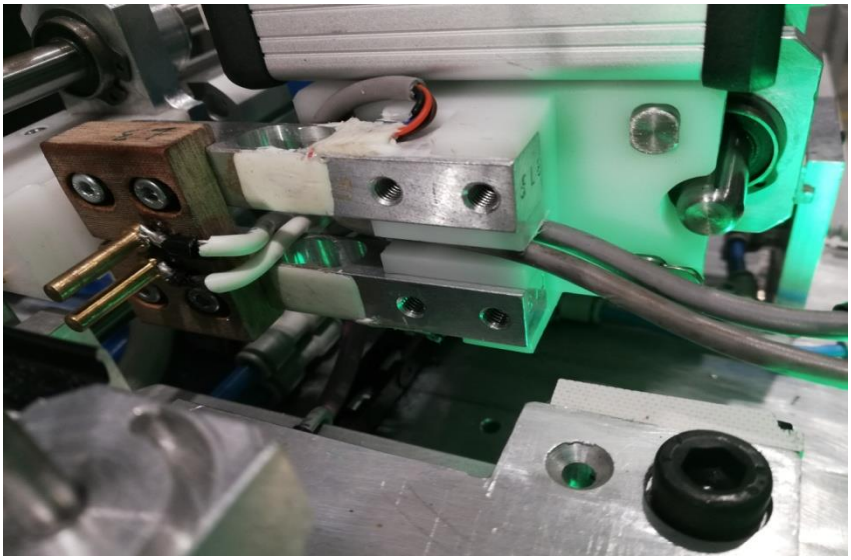


Слика 31. Тестер машина без производа.

Када се производ постави, и крене процес тестирања, кутија машине мора да буде затворена. Само тестирање и приближавање сензора до контаката и самог производа омогућавају пнеуматика као и серво мотор који се налази у положају испод производа јер он окреће положаје контаката и на тај начин, провером напона проверавамо спојеве у самом производу.

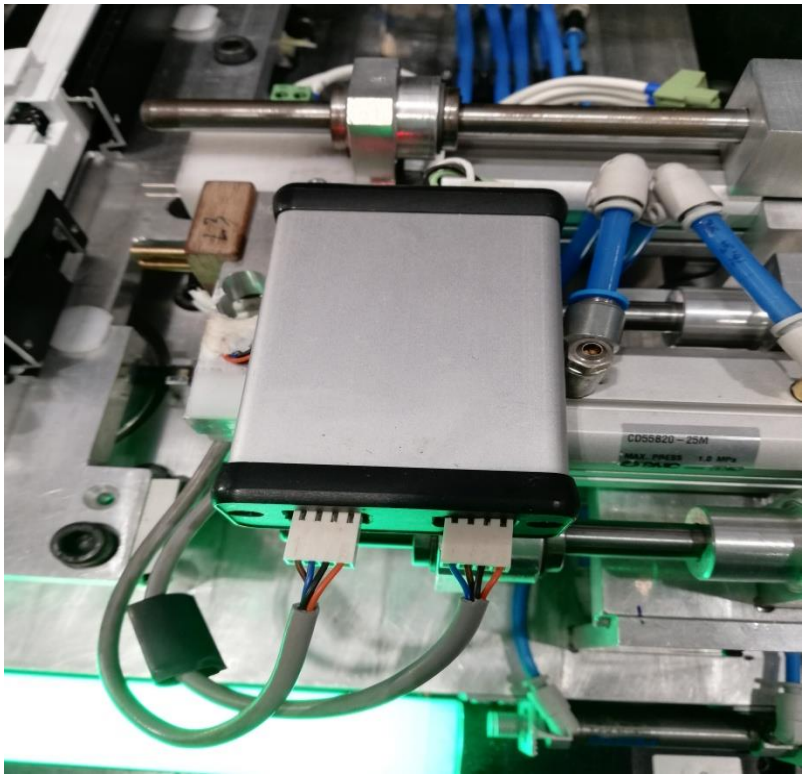


Слика 31., Тестер машина приликом тестирања производа.



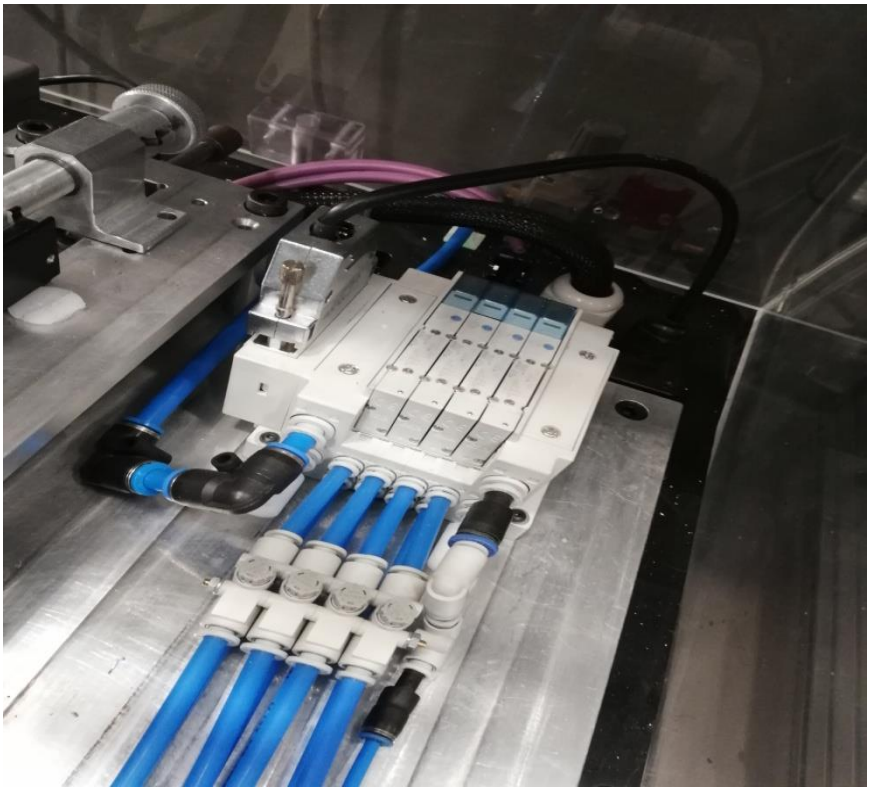
Слика 32. Сензори који преко проводника дају сигнале операционом појачавачу.

Када пнеуматика одради свој део посла, и у постављене положаје приближи сензоре, вредности са сензора које су малих нивоа морамо појачати уколико желимо да те вредности доведемо на неки од модула ПЛЦ уређаја.



Слика 33. Део машине у ком се налази електроника.

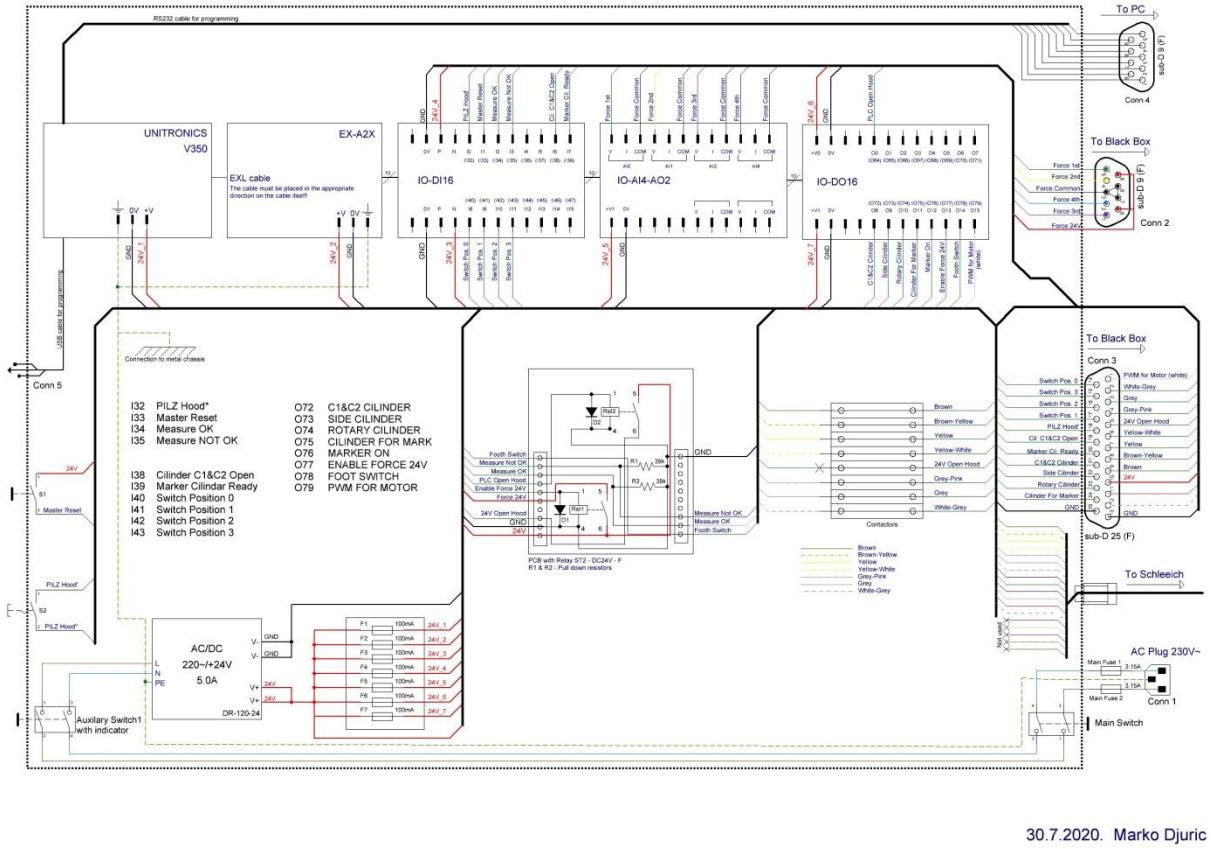
Вредности са сензора водимо на операционе појачаваче, појачавамо њихове вредности и даље их уз помоћ ЛЕДЕР дијаграма и ледер програмирања приказујемо и моделујемо у самом ПЛЦ уређају.



Слика 34. Пнеуматско острво.

На слици ц, видимо у техници названим Пнеуматско острво које је предстаља довод и развођење пнеуматике у самој машини.

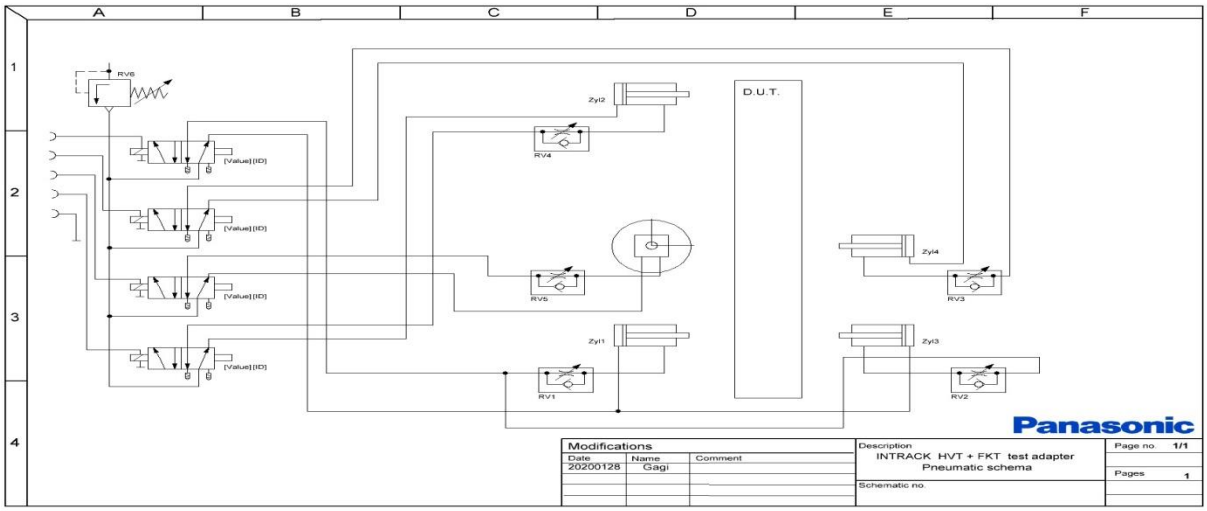
6.4 Шема ПЛЦ



Слика 35. Шема ПЛЦ уређаја.

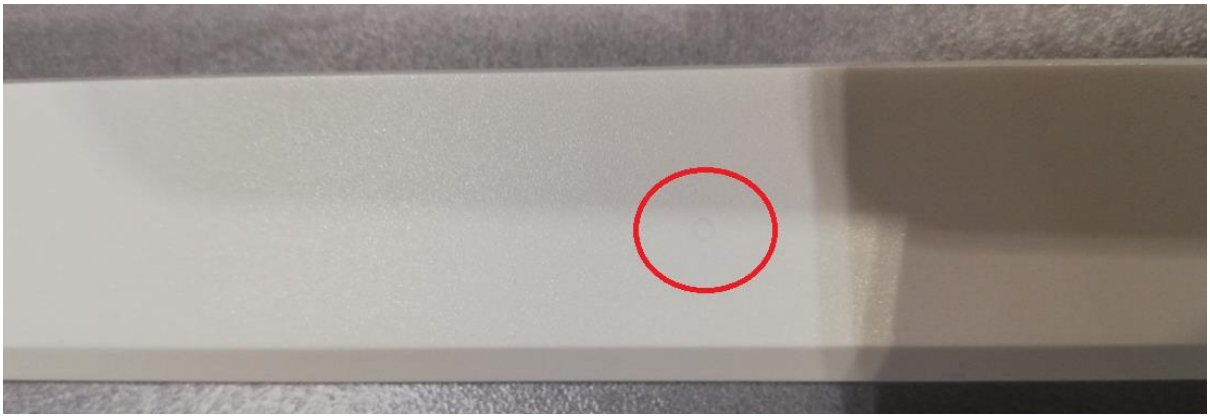
На слици 35. Приказана је шема ПЛЦ уређаја, његових модула као улазно излазних јединица.

6.4 Пнеуматска шема



Слика 36. Пнеуматска шема система.

На слици 36. Имамо приказ Пнеуматске шеме. Пнеуматика као неизоставни део овог пројекта служи да покреће и помера сензоре ка задатом положају, али и да на крају тестирања уколико је производ исправан и задовољава тест инструкције, обележи производ.



Слика 37. Обележавање исправног производа.

На слици 37. Имамо приказ обележавања исправног производа. Кружница унутар црвеном бојом означене површине означава да је производ исправан и да је задовољио тест операције и да као такав може да се појави на тржишту.

7. Опис апликације

7.1 Складиштење података на шлајху

Као што смо у тексту навели, подаци се са ПЛЦ путем мреже шаљу на шлајх. Шлајх који користимо у нашем примеру јесте модел GLP2-BASIC. Садржи заштитни кондуктор, изолован је, садржи испитивач функција као и одређену струју цурења. Са овим моделом можемо да тестирамо широк спектар електричних могућности.[11]

Слика 38. Шлајх.

Део који нас највише интересује јесте свкако како шлајх путем мреже складишти податке на серверу. Шлајху је то омогућено јер има портове који ту могућност пружају. Од портова које поседује а које можемо искористити имамо:

digital I/O, (input/output 24 V to PLC)

RS232 via SCHLEICH communication protocol

LAN via SCHLEICH communication protocol

LAN via SCPI commands

LAN via TCP/IP socket communication (individual special software required)

PROFIBUS

PROFINET

EtherCAT

CANopen

Шлајх поседује своју меморију од 4Gb која нуди довољно капацитета за готово неограничен број секвенци и резлтата теста. Резултати теста се могу приказивати и на дисплеју који поседује шлајх, такође те податке можемо преузети путем УСБ конекције а најчешће се користи чување података на серверу путем мреже.

Такав вид чувања података користимо и ми, умрежавање овог модела Шлајха. Чување податак путем мреже у нашем случају се користи јер се подаци чувају централно. Помоћу алата editor&printer који је развијен за рад са моделима овог произвођача., Бирамо централно место на ком ћемо складиштити податке.

Слика 39. Ток података до сервера.

Један од разлога зашто се користи мрежни рад јесте и испуњавање стандарда ISO 9001.

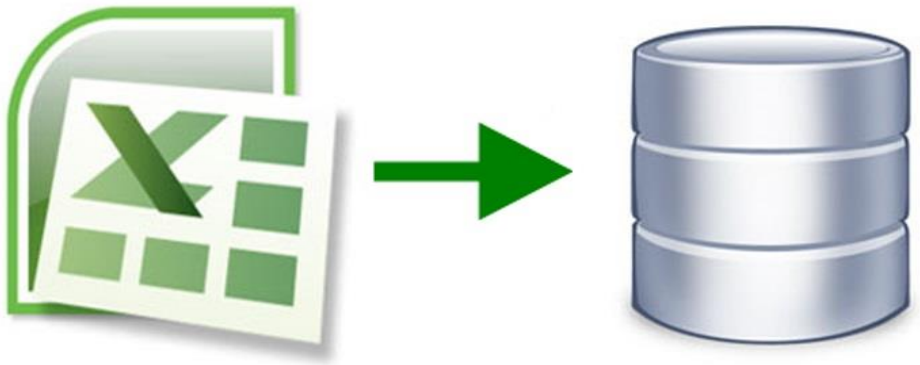
Слика 40. Програм који генерише мрежи и централном складишту податке.

На слици 40. Имамо приказ програма помоћу ког оператери и инжењери преузимају податке или те податке путем мреже шаљу на рачунар или сервер.

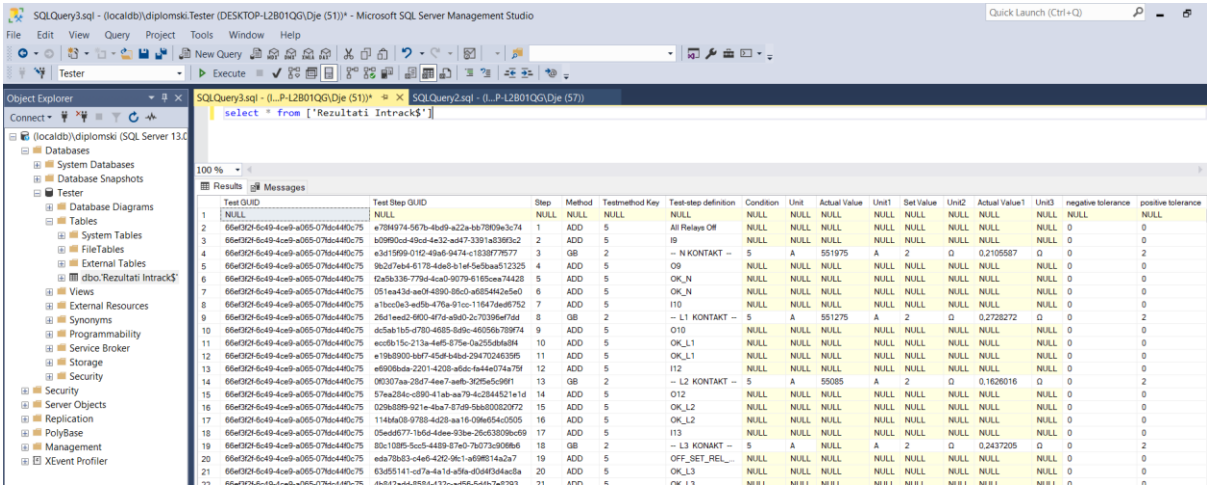
7.2 Смештање података у базу

У овом делу ћемо објаснити употребу једног информационог система у индустрији а све примењено на горе наведеном примеру тестер машине. Ова тестер машина има неколико тест инструкција. Приликом тих тест инструкција, истим управља оператер помоћу ПЛЦ уређаја. Оно што је битно јесте да ПЛЦ путем ***мреже*** шаље податке Шлајху(Уређај за високонапонско тестирање). Подаци који се на њему складиште се такође путем ***мреже*** складиште на серверу, коме могу да приступе инжењери из развојног тима и они који се баве израдом и одржавањем.

Када се подаци који се складиште у .csv фајлу, конвертују у ексел фајл, оно што ми радимо и што заправо овај рад ради јесте да ту табелу смешта у базу података.



Слика 41. Смештање података из ексел табела у базу података

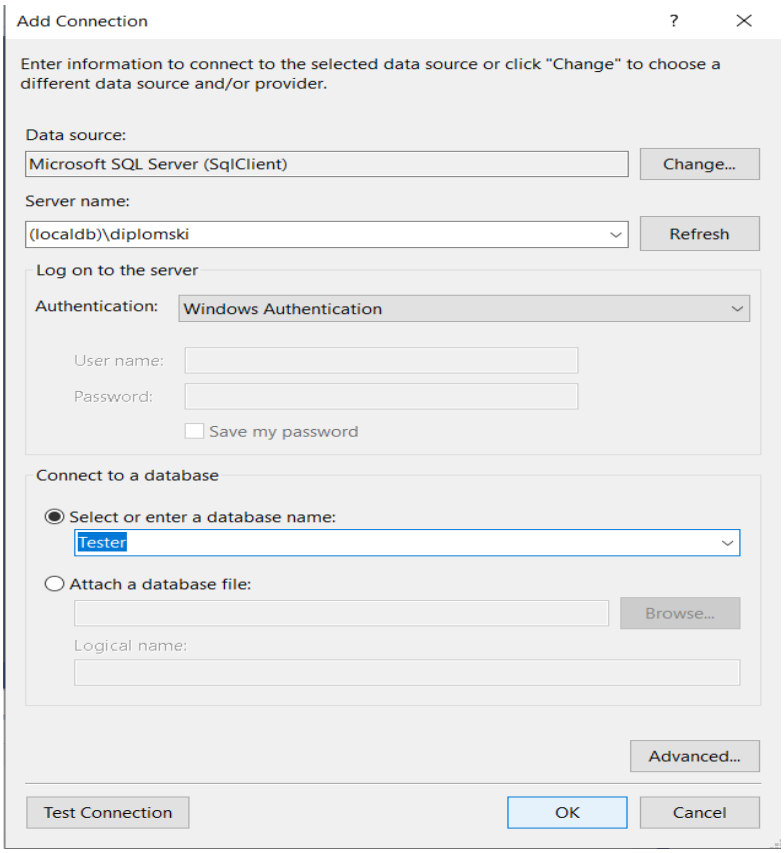


Слика 42. Смештање података из ексел табела у базу података.

На слици 42. Имамо приказ базе података у развојном окружењу Microsoft SQL Server Management Studio.

7.3 Повезивање базе и апликације

Након што смо направили базу података од података које смо преузели са мреже и шлајха, тежимо ка томе да повежемо базу података са апликацијом. Па зато, с обзиром да апликацију моделујемо у развојном окружењу Microsoft Visual Studio, приступамо повезивању базе и апликације.



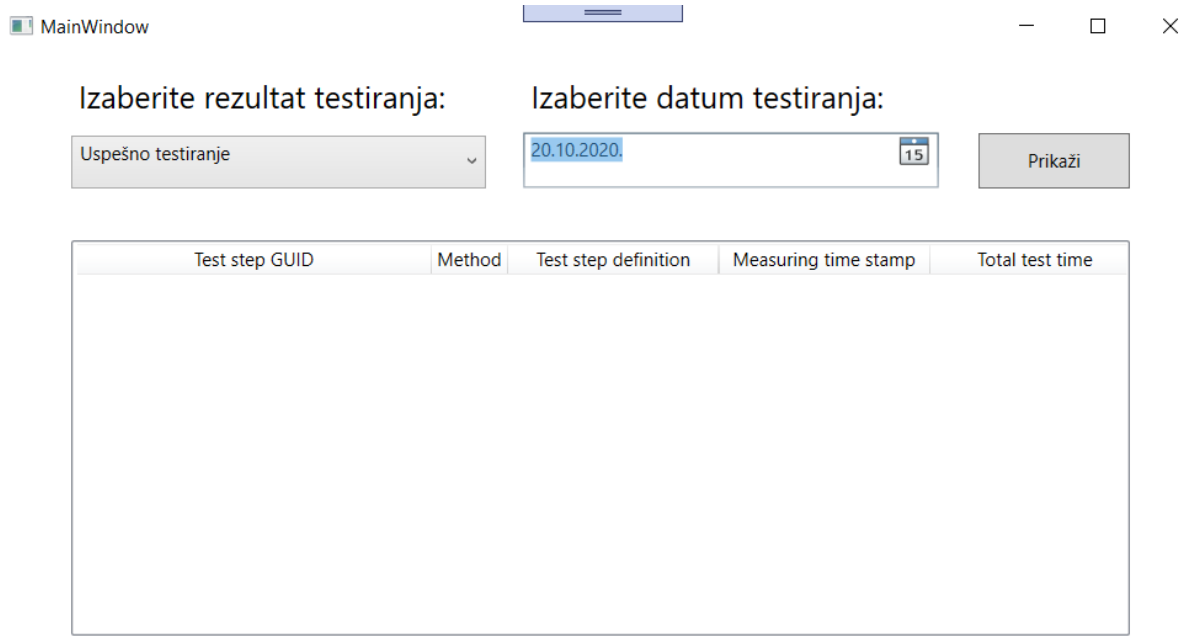
Слика 43. Повезивање базе са апликацијом.

На слици 43. Видимо приказ конектовања апликације на базу података, тако што се прво конектује на сервер а потом и на одговарајућу базу података. Након што одрадимо конекцију, можемо приступити изради информационог система.

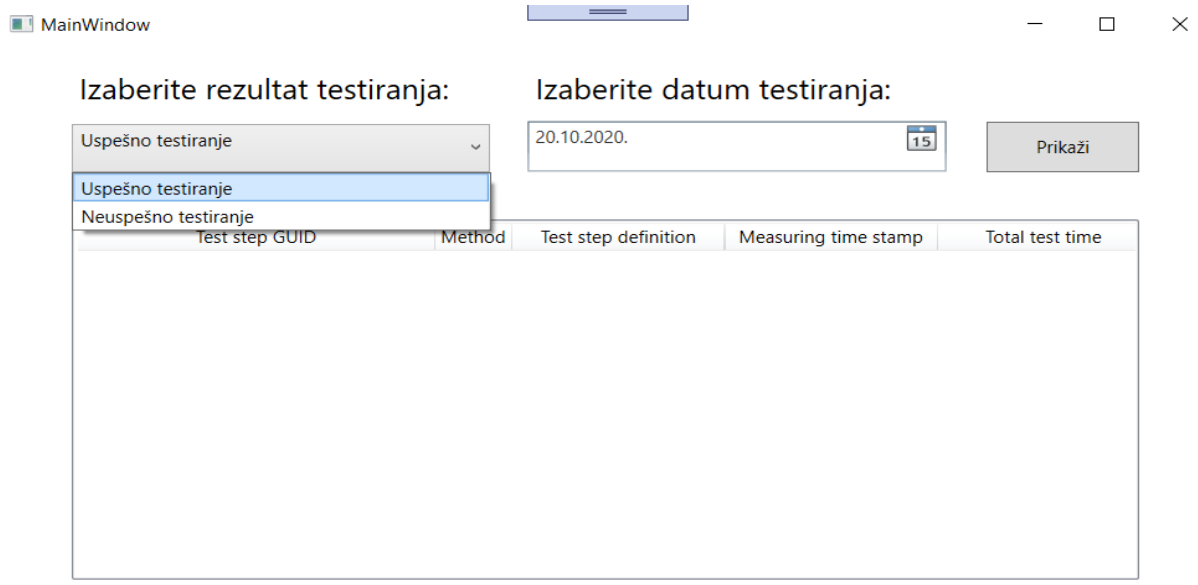
7.4 Апликација

Апликација као таква замишљена је да првенствено пружи бољи увид и даје приказ тестираних уређаја. Оно што је од највеће важности јесте то да пре свега, инжењери који рукују овом апликацијом имају брз приказ тестираних уређаја и то оних који су тестирани успешно, као и оних који су тестирани неуспешно. Зато ова апликација нуди претраживање базе на основу два параметра. Према томе да ли је производ прошао тест или не, и по томе ког датума се вршило тестирање.

Карактеристика оваквих података је таква да се они не могу мењати, додавати и брисати. Зато је ова апликација искључиво карактера прегледа и служи да инжењери и оператери који би је користили у што бржем року добију њима битне информације о производу на основу параметара које су унели.

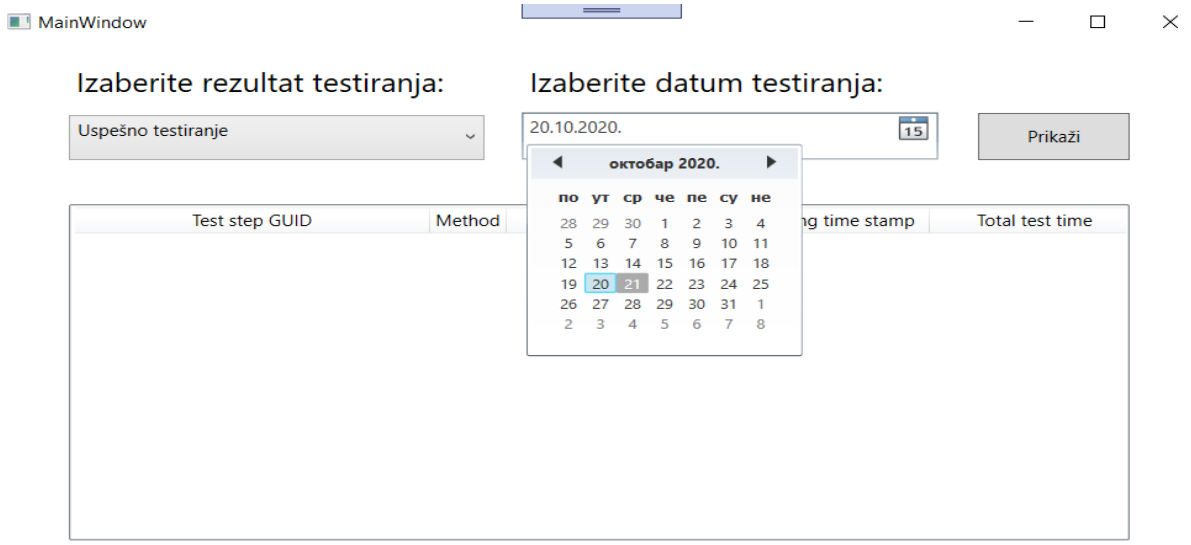


Слика 44. Почетни приказ апликације.



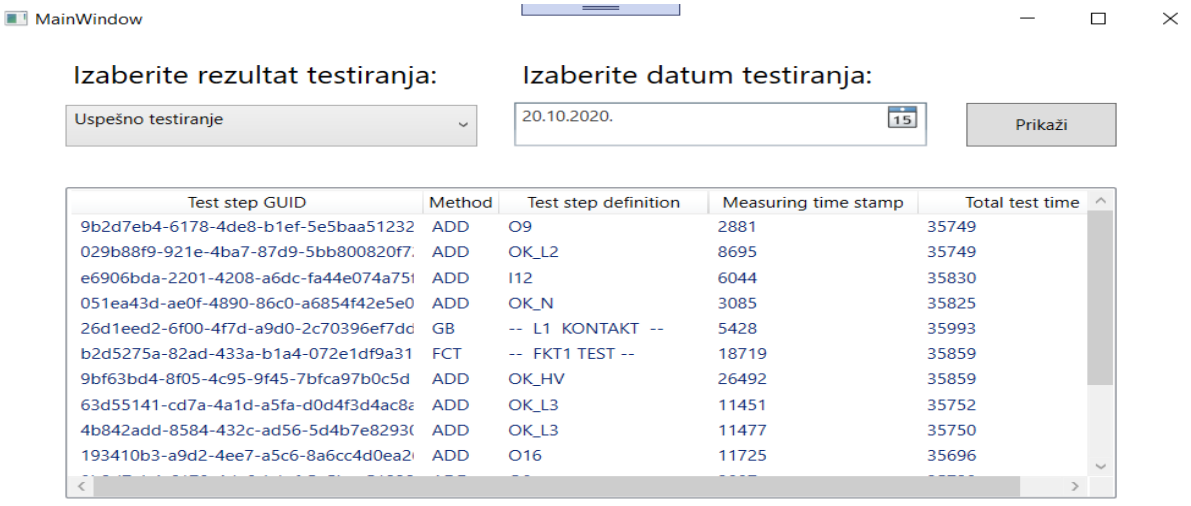
Слика 45. Одабир производа на основу исхода тестирања.

Претрага базе података на основу исхода тестирања је веома битно из неколико погледа. Пре свега, пошто је велики број тестирања, не коришћењем, односно класичним ексел претраживањем би сваког пута губили доста времена у претрази уколико желимо да пронађемо производ који је пао на тестирању. Фокус је свакако на производима који су пали тестирање јер на основу њихових параметара можемо да закључимо одређене појединачности самог производа као и да покушамо да нађемо узрок пада на тестеру.

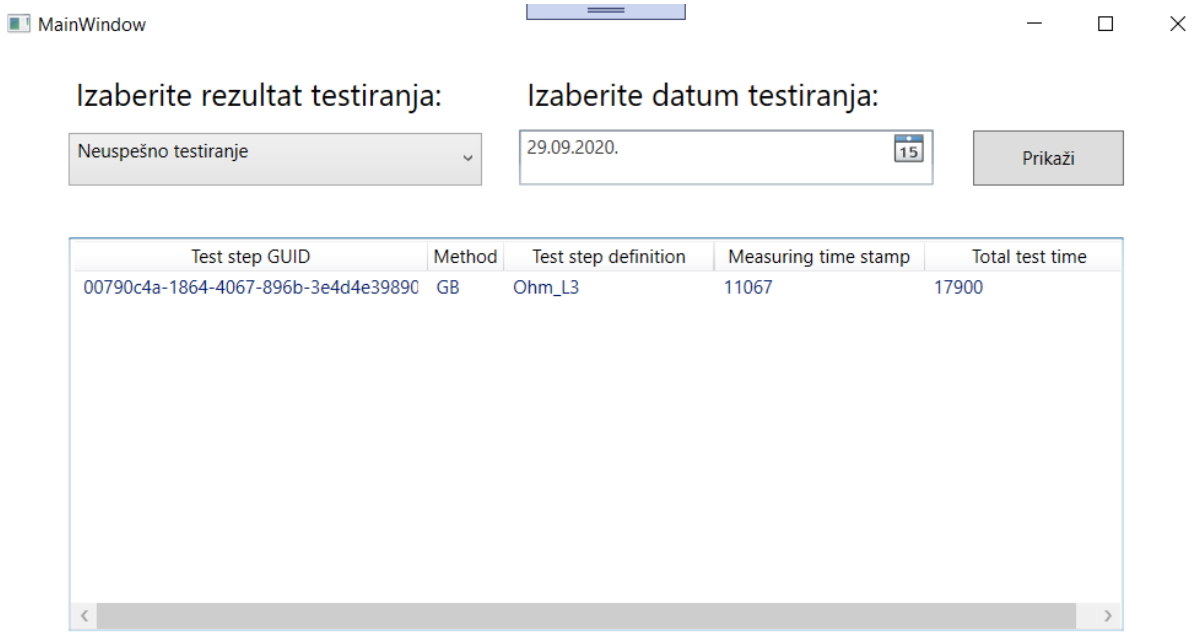


Слика 46. Одабир производа на основу датума тестирања.

На слици 46. Видимо приказ одабира производа и претрагу базе података на основу датума тестирања. Разлог зашто смо ово узели као параметар претраге лежи у томе да на основу датума имамо увид у то ко је радио на тестер машини. Разлог овог начина јесте да имамо ближи увид у то ко је радио на тестер машини, такође можемо повезати евентуалне лоше резултате и са серијом из које долазе производи па да самим тим унапредимо и отклонимо недостатке на конкретној серији.



Слика 47. Приказ података успешно тестираног производа



Слика 48. Приказ података неуспешно тестираног производа

На слици 48. Видимо приказ података неуспешно тестираног производа. Оно што видимо на листи података јесте пет колона.

Прва колона представља ознаку појединачног тестирања и генерише се за свако тестирање.

Друга колона представља метод начин тестирања.

Трећа колона представља корак у тестирању, пошто смо раније говорили да током тестирања производ пролази кроз неколико фаза тестирања.

Четврта колона представља време у ком се прекинуло тестирање.

Пета колона представља укупно време тестирања.

8. Закључак

Након свега наведеног може се закључити и видети директна примена и предности које један информациони систем омогућава једном индустријском постројењу. Његове функције и делатности у најбољем светлу виде и користе радници међутим поред радника у оваквим делатностима, овакви системи у многоме олакшавају посао и инжењерима, јер имају увид и могу да добију информацију и путем података евентуално побољшају функционалност машине или производног процеса уопштено. Свакако примена информационих система у индустрији даје велики увид у продуктивност рада па можемо закључити да су подаци који се добијају на основу рада машине и оператера драгоцени. Оно што сваку индустрију и фабрику интересује јесте свкако економски аспекат, прикупљање и процесуирање оваквих података такође може олакшати и дати бољи приказ и економском делу компаније. Долази се до закључка да су овакви системи данас доведени до савршенства уколико оно постоји, јер у само пар кликова долазимо до информација о производима. Очекује се да ће систем који је развијен уз потребну надоградњу наћи примену у неком процесуирању података и унапређивању производног процеса, тестирања истог а самим тим и подићи квалитет производа.

9. Литература

[1] Лазаревић Б.: Базе података, ФОН Београд, Београд 2003.

[2] Павловић-Лажетић Г.: Основе релационих база података, Математички факултет, Београд, 2000.

[3] R. Elmasri, S. Navathe, Fundamentals of Database Systems, Addison-Wesley, Boston, 2003.

[4] Преузето: http://www.ibexpert.net/ibe/uploads/Doc/all.html датум приступа 22.09.2020.

[5] Драган М. Маринковић: Програмабилни логички контролери – увод у програмирање и примену

датум приступа 19.10.2020.

[6] Преузето: http://elektronika.elfak.ni.ac.rs/_FILES/sandra.djosic/4%20plc.pdf

датум приступа 19.10.2020.

[7] Преузето: https://www.yumpu.com/xx/document/view/18138636/programiranje-industrijskih-kontrolera/ датум приступа 19.10.2020.

[8] Преузето: https://www.ucg.ac.me/skladiste/blog_6146/objava_22904/fajlovi/PLC.pdf датум приступа 18.10.2020

[9] Преузето: https://www.vossloh-schwabe.com/ датум приступа 20.10.2020

[10] Преузето: https://www.globalsources.com/LED-light/LED-TRACK-LIGHT-1171083751p.htm#1171083751 датум приступа 20.10.2020

[11] Преузето: https://www.schleich.com/en/ датум приступа 21.10.2020