МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Б. В. Орловський

МЕХАТРОНІКА В ГАЛУЗЕВОМУ МАШИНОБУДУВАННІ

Навчальний посібник

Рекомендовано Вченою радою Київського національного
університету технологій та дизайну для студентів денної,
заочної та дистанційної форм навчання
за галуззю знань «Механічна інженерія»
спеціальностей 133 «Галузеве машинобудування»
та 131 «Прикладна механіка»

Київ 2018

УДК 67/68.05:621.865.8]:004.9(075.8)
О-66

Р е ц е н з е н т и:

Скиба М. Є. – д-р техн. наук, проф., ректор Хмельницького національного університету;

Губарев О. П. – д-р техн. наук, проф., професор кафедри прикладної гідроаеромеханіки і мехатроніки НТУУ «КПІ імені Ігоря Сікорського»;

Якимчук М. В. – д-р техн. наук, проф., професор кафедри мехатроніки та пакувальної техніки Національного університету харчових технологій.

Рекомендовано Вченою радою Київського національного університету технологій та дизайну як навчальний посібник для навчання студентів денної, заочної та дистанційної форм навчання за галуззю знань «Механічна інженерія», спеціальностей 133 «Галузеве машинобудування» та 131 «Прикладна механіка» (Протокол № 3 від 29 листопада 2017)

Орловський Б. В.

О-66 Мехатроніка в галузевому машинобудуванні: навчальний посібник / Б. В. Орловський. – К.: КНУТД. – 2018. – 416 с.

ISBN 978-617-7506-11-8

Навчальний посібник містить програмний матеріал, який передбачений програмою дисципліни «Мехатроніка в галузевому машинобудуванні» для студентів спеціальностей «Галузеве машинобудування», «Прикладна механіка», рівня вищої освіти – *бакалавр* та студентів спеціальності «Галузеве машинобудування», які навчаються за освітньою програмою «Обладнання легкої промисловості та побутового обслуговування» при вивченні програмного матеріалу дисципліни «CAM-технології комп'ютерно-інтегрованого обладнання», рівня вищої освіти – *магістр*.

УДК 67/68.05:621.865.8]:004.9(075.8)

ISBN 978-617-7506-11-8

ВСТУП

Навчальні посібники на відміну від підручників, як правило, видаються згідно програм навчальних дисциплін і містить матеріали, в основному, які базуються на матеріально-технічної базі для лабораторних робіт у тому навчальному закладі, у якому працює автор. Це підтверджує і навчальні посібники [1,2,3], які включають оригінальні матеріали і матеріали міжвузівських студентських олімпіад з однойменних навчальних дисциплін і науково-практичних конференцій за тематикою з мехатроніки. Ведучою кафедрою з мехатрониці на Україні є кафедра прикладної гідроаеромеханіки і механотроніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», на якої щорічно проводяться міжвузівська студентська олімпіада «Мехатроніка галузевого машинобудування» під науковим керівництвом професора Губарева Олександра Павловича.

В навчальному посібнику розглянути питання аналізу і синтезу мехатронних систем в застосуванні до механізмів технологічних машин і механіко-технологічних систем легкої промисловості.

Навчальний посібник «Мехатроніка в галузевому машинобудуванні» містить програмний матеріал, який передбачений навчальної і робочій програмами дисципліни «Мехатроніка в галузевому машинобудуванні» для студентів денної, заочної та дистанційної форм навчання з галузі знань «Механічна інженерія», спеціальності «Галузеве машинобудування», «Прикладна механіка», ступеня вищої освіти – бакалавр. Матеріали навчального посібника також можуть бути використаний при вивченні навчальної дисципліни «САМ-технології комп’ютерно-інтегрованого обладнання» для студентів спеціальності «Галузеве машинобудування» за освітньою програмою «Обладнання легкої промисловості та побутового обслуговування» ступеня вищої освіти – магістр.

В навчальному посібнику розглянути принципи об'єктно-орієнтованого підходу до побудови механіко-технологічних систем на основі типових мехатронних модулів. Розглянуті САПР-побудова і аналіз роботи схем циклових мехатронних систем в програмному середовищі. Наведені правила аналізу технологічних графів для мехатронних систем с БІстабільним і МОНОстабільним керуванням для наступного їх застосування при переході від рівнянь причинно-спадкових зв'язків до складання схем керування без контролера і з контролером. Наведені приклади побудови функціональних графів, систем рівнянь причинно-

спадкових зв'язків і програм їх реалізації з використанням програмованого логічного контролера при створенні проектів в програмному середовищі для комп'ютерно-інтегрованих технологічних машин легкої і текстильної промисловості з важільними і кулачковими механізмами.

Автор висловлює вдячність рецензентам професору М.Є. Скибі, професору О.П. Губареву і професору М.В. Якимчуку за зроблені зауваження і доповнення та за конструктивну співпрацю по впровадженню в Київському національному університеті технологій та дизайну організаційних і методичних положень з мехатроніки в галузевому машинобудуванні легкої промисловості на кафедрі машин легкої промисловості зараз кафедра прикладної механіки та машин.

1. СТРУКТУРА І ЕЛЕМЕНТИ МЕХАТРОНІКИ

1.1. Компетентності з дисципліни «Мехатроніка в галузевому машинобудуванні»

Сучасне наукоємне галузеве машинобудування базується на теоретичних основах механіки, електроніки, прикладному програмуванні та математики. Для інженера-механіка практичне застосування цих наук засноване на залишку інформації у відповідності зі схемою на рис.1, де наведені відсоток засвоюваних знань і отриманих компетенцій з навчальних технічних дисциплін при застосування сучасних методів навчання типу «STEM-освіта».

Рис. 1. Схема прямих і зворотніх зв'язків при STEM-освіті, як основи навчання з дисципліни «Мехатроніка в галузевому машинобудуванні»

STEM-освіта – абревіатура англійських слів «**S**cience, **T**echnology, **E**ngineering, **M**athematics («**Н**аука, **Т**ехнологія, **І**нженерія, **М**атематика»), яка заснована на модерації. *Модерація* - це метод оптимізації навчання

з точки зору мінімізації затрат часу і зусиль, який сприяє оптимізації ідей, думок, поглядів і бачення для отримання компетенцій (знань, навиків і вмінь) студентами. За допомогою модерації *лектор-тренер* як би «дістає» з учасників їх погляди і з певного питання. Важливими інструментами модерації є *візуалізація навчання* і ***навчання засобами моторики*** («*зроби сам, своїми руками*»), як це зображено на рис. 1.

Мета вивчення навчальної дисципліни «Мехатроніка в галузевому машинобудуванні» є отримання компетенцій, в тому числі вмінь і навичок (*skills*) вирішувати завдання, які пов'язані з особливостями побудови, експлуатації, проектуванні (аналізу і синтезу) мехатронних циклових систем різних рівнів складності [2] та їх зборки і наладки у складі технологічного обладнання. В навчальному посібнику вперше розглядаються приклади застосування мехатроніки до автоматизації механізмів машин легкої промисловості.

Головним *завданням* дисципліни «Мехатроніка в галузевому машинобудуванні» є формування сучасних інженерів-механіків, які володіють міждисциплінарними зв'язками знань з різних галузей технічних наук, які пов'язані з комп'ютерними технологіями на виробництві. *Мехатроніка* об'єднує механіку, електроніку, пнемо- та гідроавтоматику і програмування контролерів, вбудованих в механіко-технологічні системи і технологічні машина галузі.

Дисципліна базується на раніше вивчених загальнотеоретичних та загально-інженерних дисциплінах і є базовою при вивченні студентами наступних професійно-орієнтованих дисциплін спеціальності: «Механічна технологія і обладнання легкої промисловості», «Проектування швейних машин», «Проектування в'язальних (трикотажних) машин», «Проектування взуттєвих машин».

У результаті вивчення навчальної дисципліни студент повинен:

знати: можливості і особливості застосування циклових систем для автоматизації технологічних машин галузей легкої промисловості (швейних, в'язальних і взуттєвих) та апаратів побутового обслуговування;

вміти: складати і налагоджувати схеми пневмоавтоматики і пнемо-електроавтоматики для автоматизації технологічних машин галузі.

1.2. **Структура і місце мехатроніки у розвитку комп'ютерних обчислювальних систем**

Сформульована мета навчальної дисципліни досягається вирішуванням наступних трьох основних задач:

1 задача - задача механіки направлена на спрощення кінематики просторових і багато важільних механізмів машин легкої промисловості за рахунок використання технічних засобів мехатроніки при створенні виконавчих механізмів і машин з електронною пам'яттю і «інтелектом»;

2 задача - задача електроніки направлена на вивчення на засадах об'єктно-орієнтованого проектування метода графів і технічних засобів для реалізації прямих і зворотних зв'язків в енергетичних і в інформаційних каналах передачі сигналів команд керування, сигналів адресів і иигналів даних в комбінованих схемах з контролером;

3 задача - задача інформатики направлена на використовування знань інформаційних технологій для розкриття внутрішніх і зовнішніх зв'язків у тріаді «*технологічна машина – граф і рівняння причино-наслідкових зв'язків циклової системи керування - програмований контролер*».

До 50% ринкової вартості сучасних технологічних машин легкої промисловості при їх виготовленні складає «залізо» машин із жорсткою системою керування типу «*розподільний вал*». Ще 50% вартості у вартості машин і механіко-технологічних систем складає вбудовані в «залізо» електронні, електромеханічні та програмні технічні засоби гнучкої системи керування виконавчими механізмами, які кінематичне не з'єднані з головним валом машини. Такий стан обумовлює і зміну вимог на ринку праці до випускників за спеціальністю «Галузеве машинобудування» та спеціальністю «Прикладна механіка», які повинні мати компетенції з механіки технологічних машин, електромеханіки, електроніки та спеціального прикладного програмування для автоматизації технологічних машин і механіко-технологічних систем.

Місце мехатроніки у структурі динаміки розвитку продуктивності комп'ютерних обчислювальних систем наведено на рис.1.1, де прийняті наступні скорочення:

PC – персональні комп'ютери (**P**ersonal **C**omputer);

WS – робочі станції (**W**ork **S**tation);

SC – суперкомп'ютери (**S**uper **C**omputer);

CS – кластерні системи (**C**luster **S**ystems).

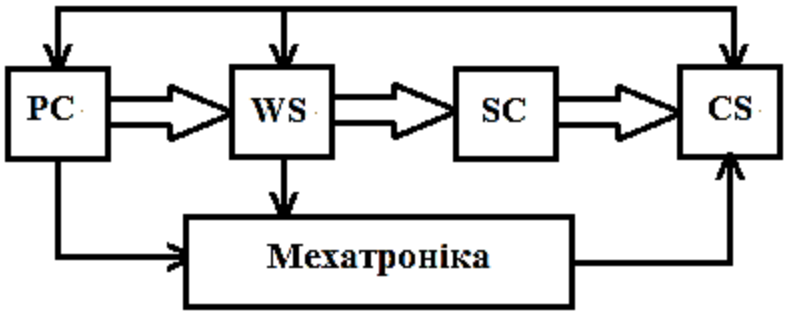


Рис. 1.1. Структурна схема розвитку комп'ютерних обчислювальних систем і їх зв'язок з мехатроникою

Персональні комп'ютери – успадковують, з точки зору об'єктно-орієнтованого проектування, контролери або контролери є нащадком персональних комп'ютерів (**PC** – **P**ersonal **C**omputer). При цьому мається на увазі однопроцесорна архітектура контролера фірми виробника або однопроцесорні системи PC на платформі Intel, AMD та інших, що працюють під керуванням одного користувача операційної системи MS Windows та ін.

Робочі станції – успадковують, з точки зору об'єктно-орієнтованого проектування, персональні комп'ютери або персональні комп'ютери є нащадком робочих станцій (**WS** – **W**ork **S**tation). При цьому PC можуть мати декілька процесорів з RISC-архітектурою для багато поточної операційної системи UNIX. Контролери робочих станцій підтримують віддалений доступ до датчиків і виконавчих механізмів механіко-технологічних і технічних систем підприємств галузі. Можуть обслуговувати обчислювальні потреби невеликої групи користувачів.

Суперкомп'ютери – успадковують, з точки зору об'єктно-орієнтованого проектування, робочі станції або робочі станції є нащадком суперкомп'ютерів (**SC** – **S**uper **C**omputer). Суперкомп'ютери поки мають екзотичний характер і разові приклади застосування тому, що надзвичайно дорогі і будуються на векторно-конвеєрної архітектурі процесорів. Така архітектура процесорів дозволяє реалізувати конвеєрну організацію команд програми з векторними операціями над масивами даних.

Кластерні системи – це дешева альтернатива суперкомп'ютерам, а тому успадковують, з точки зору об'єктно-орієнтованого проектування,

і персональні комп'ютери, і робочі станції або персональні комп'ютери і робочі станції є нащадком кластерних систем (**CS** – **C**luster **S**ystems).

Поширюючи думку авторів [1] «персональные компьютеры смогли стать персональными не только для людей, но и для машин» з наведеної структурної схеми на рис. 1.1 випливає також те, що робочі станції (**WS**) стають «робочими» підприємств та систем, а такі підприємства та системи утворюють кластерні системи (**CS**).

Структура мехатроніки і її зв'язок з інженерними поняттями і дисциплінами наведені на рис. 1.2.

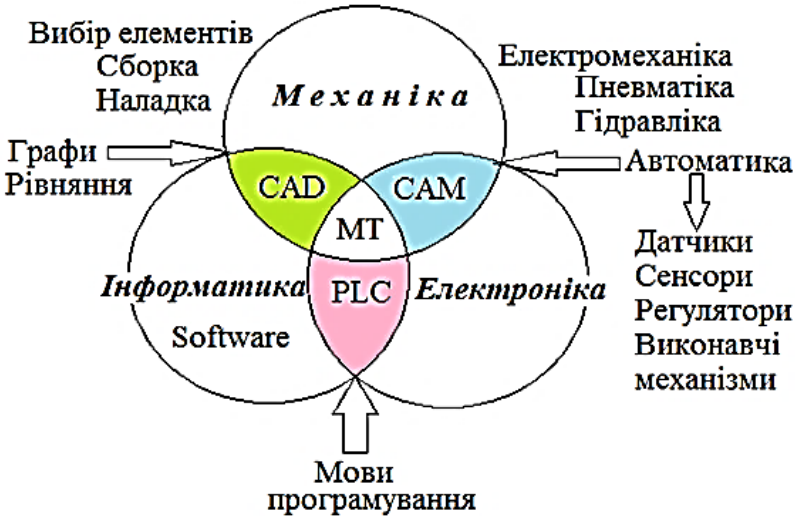


Рис. 1.2. Структура мехатроніки і зв'язок дисципліни з інженерними поняттями

На перетині множин «***Механіка***» та «***Електроніка***» утворюється множина **CAM**-технологій (скорочення *від англ.* **C**omputer-**A**ided **M**anufacturing - «Автоматизоване виробництво») – це програмні і комп'ютерні апаратні засоби для автоматизації виробництв. CAM-програми реалізують геометрічні 3D-моделі деталей з різних матеріалів і які створені в **CAD**-програмах для їх наступного виготовлення на **CNC**-машинах, **CNC**-верстатів або на 3D-принтерах.

На перетині множин «***Електроніка***» та «***Інформатика***» утворюється множина програмованих логічних контролерів **PLC** (**P**rogrammable **L**ogic **C**ontroller).

На перетині множин «***Інформатика***» та «***Механіка***» утворюється множина **CAD** (**C**omputer-**A**ided **D**esign - «Проектування за допомогою ЕОМ») - це скорочення для позначення автоматизованих систем проектування з використанням комп'ютерних технологій. Як правило, CAD-системи застосовують для тривимірного геометричного моделювання та дизайну інженерних виробів, а також оформлення проектної (конструкторської) документації.

Вивчення, використання і отримання CAD-компетенцій пов’язане з наступними дисциплінами: «Інформатика», «Електротехніка, електроніка та мікропроцесорна техніка», «Основи автоматики і автоматизації технологічних процесів та машин», «Основи САПР в галузевому машинобудуванні», «Автоматизоване проектування обладнання легкої промисловості» та іншими.

На перетині множин «***Електроніка***» і «***Інформатика***» утворюється множина програмованих логічних контролерів **PLC** (**P**rogrammable **L**ogic **C**ontroller).

PLC, поширена назва **ПЛК** – це електронний пристрій, який використовується для автоматизації технологічних машин таких як, швейні, в’язальні, взуттєві та інші машини легкої і текстильної промисловості. **ПЛК**, це спеціалізований комп'ютер реального часу, що розроблений на основі мікроконтролера. Основною його відмінністю від персонального комп’ютера є наявність портів вводу (***Input*)** для датчиків і портів виводу (***Output***) для виконавчих механізмів, а також можливість надійної роботи у виробничих умовах і при різних умовах: широкий діапазон температур, висока вологість, сильні електромагнітні завади, вібрації і т. п.

Вивчення архітектури **ПЛК** для бакалавра-механіка пов’язане з вивченням наступних дисциплін навчального плану: «Інформатика», «Електротехніка, електроніка та мікропроцесорна техніка», «Основи автоматики і автоматизації технологічних процесів та машин» та інших.

Об’єднання трьох множин «***Механіка***», «***Електроніка***» та «***Інформатика***» ***(«Програмне забезпечення»)*** утворює інтегроване поняття і зміст «***Мехатроніка***», яке позначено скороченням «**МТ**» на рис. 1.2.

При проектуванні автоматизованих технологічних машин персональний комп'ютер "переноситься" з офісу у виробничий в цех, де використовується як програматор для контролера вбудованого в технологічну машину і який за допомогою своїх портів вводу/виводу має прямі і зворотні зв'язки з датчиками і виконавчими механізмами технологічної машини.

Реалізації CAM-технологій на виробництві відбувається шляхом застосування комп'ютерне-інтегрованого обладнання, узагальнена блок-схема якого наведена на рис.1.3. На схемі наведено позначення **CNC** (сі-єн-сі) «**C**omputer **N**umerical **C**ontrol» – це англомовне скорочена назва верстатів машинобудування та технологічних машин легкої промисловості з **ЧПК (Ч**исловим **П**рограмним **К**еруванням). Поширена скорочена назва **ЧПУ** (*рос*. - **Ч**исловое **П**рограмное **У**правление).

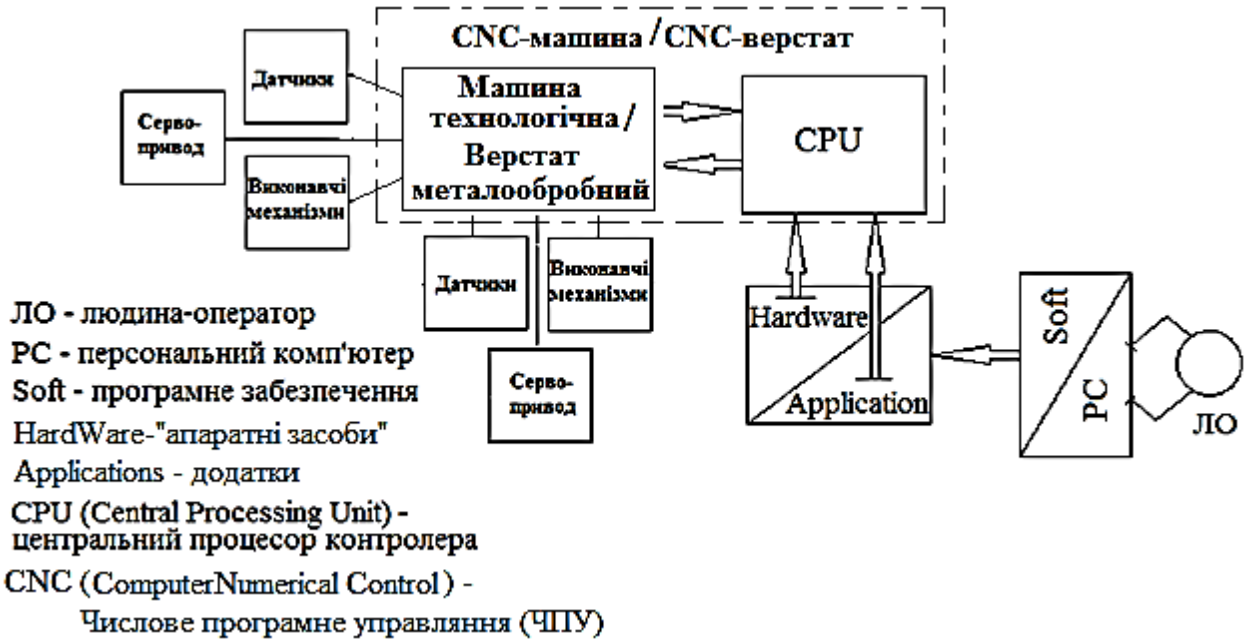


Рис. 1.3. Узагальнена блок-схема комп'ютерне-інтегрованого обладнання

Механічна, технологічна, електромеханічна, електронна і програмна складові мехатроніки у сучасних технологічних машинах галузей легкої промисловості наведені на рис. 1.4, де прийняти наступні умовні скорочення:

ТМ – технологічна машина;

СУ – система керування ;

ПР – промисловий робот;

ЕД – електродвигун;

ЧПК – числове програмне керування ;

ПЛК – програмований логічний контроллер;

ГАМ – гнучкий автоматизований модуль;

ГАЛ – гнучка автоматизована лінія;

ГАВ – гнучке автоматизоване виробництво;

КІТ МЛП – **К**ом'ютерно-**І**нтегровані **Т**ехнологічні **М**ашини **Л**егкої **П**ромисловості.

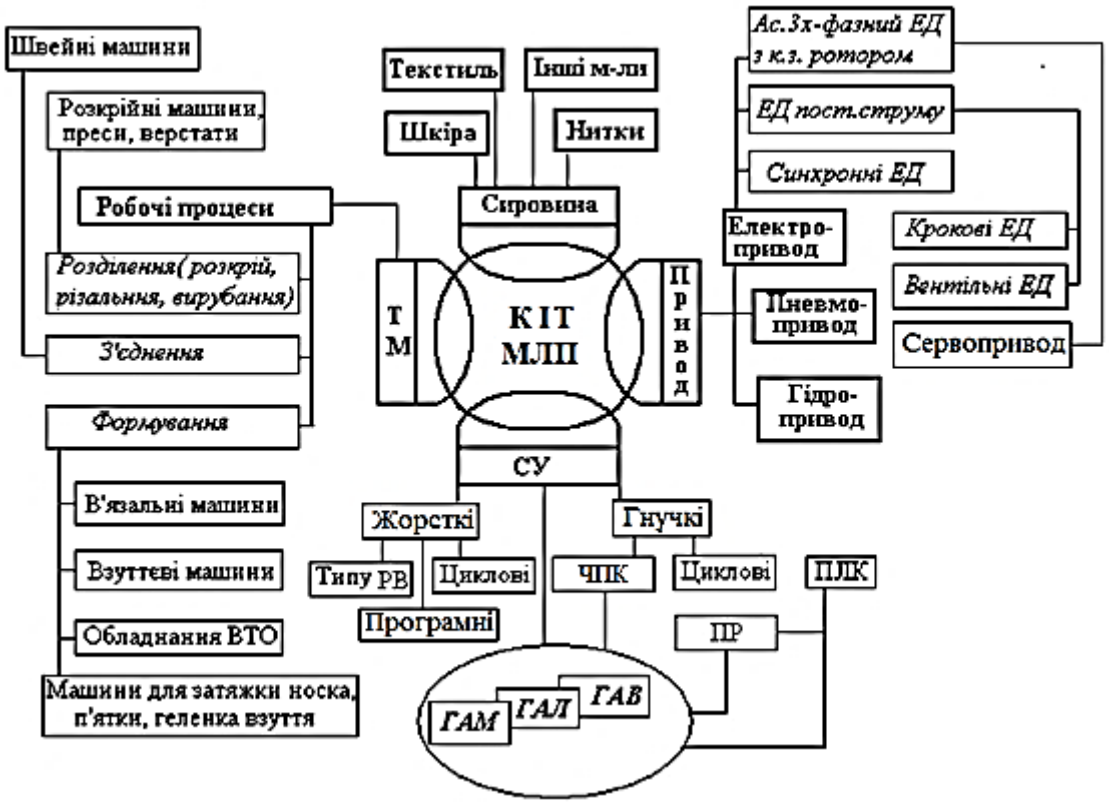


Рис. 1.4. Структурна схема Ком'ютерно-Інтегрованих Технологічних Машин Легкої Промисловості (**КІТ МЛП**)

Ком'ютерно-Інтегровані Технологічні Машини Легкої Промисловості мають в структурі складові («залізо» та текстильні матеріали – нитки і тканини або трикотаж), які людина бачить своїм зором і складові мікросхем (наприклад, регістри, транзистори, резистори та інші), які реально існують в машині у вигляді електронних модулів на

друкованих платах, але людина не бачить їх складові своїм зором. Не бачить своїм зором людина також і машинні коди програмно-інформаційного забезпечення сучасних технологічних машин при їх роботі. Все чого не бачить людина своїм зором при роботі з машинами є узагальненою складністю ком'ютерно-інтегрованих машин і технічних систем при їх проектуванні, виготовленні, сервісному обслуговуванні і ремонті.

Дійсно, людина бачить зором без застосування додаткових засобів для зору ланки механізмів, механізми і пристрої машини, тобто все, що зроблено в машині за традиційною технологією машинобудування і приладобудування з матеріалів за принципом "*деталь - складальна одиниця*". Це відноситься до деталей, які виготовляються із заготівок з використанням традиційних операцій машинобудування таких, що потребують **операцій розділення** (наприклад, операції *різання*), **з'єднувальних операцій** (наприклад, *зварювання*, зборка рухомого з'єднання та інші) і **операцій формо утворювання** (операції *формування*).

Аналогами таких технологічних операцій у виробництвах легкої промисловості, наприклад у швейному виробництві це операції: **розкроювання** текстильних або шкіроподібних матеріалів і отримання деталей крою, **з'єднання** деталей крою нитковим способом на швейних машинах нитковим способом або безнитковим способом, **формування** виробу способом волого-теплової обробки на пресах або паро-манекенах.

Те, що відбувається в електронних елементах і програмних компонентах машини людина не бачить безпосередньо своїм зором, а розуміє про їх роботу по кінцевому результату - наприклад, переміщеннями технічних об'єктів від виконавчих програмно-керованих механізмів. При цьому «деталями» є атоми, «*p-n*» і «*n-p*» переходи в кристалі напівпровідникових матеріалів і *біти* («1» і «0») *адресів, команд і сигналів керування* програмного забезпечення мехатроніки. Такі *технології знизу-вверх* виготовлення виробів з «деталей» базуються не на технологіях виділення певних об'ємів матеріалів, які становлять припуск для створення фізичної форми виробу, а на технологіях пошарового нарощуванні об'єктів до досягнення необхідних їх характеристик на нано-, микро- і макрорівнях при конструюванні і отриманні фізичної поверхні потрібної форми деталі.

Такими є також реалізація інформаційних *технологій знизу-вверх* і які відбуваються *від непрозорих для людського ока* машинних кодів команд програмного забезпечення *до прозорих для людського ока* механічних

програмованих рухів (траєкторій) робочих інструментів механізмів технологічних машин і робочих інструментів верстатів машинобудування.

1.3. **Структурно-логічні зв'язки мехатроніки з фундаментальними і інженерними дисциплінами**

З рис. 1.2 випливають наступні зв'язки навчальної дисципліни «Мехатроніка в галузевому машинобудуванні» з інженерними дисциплінами.

З CAM-технологіями пов'язане науково і виробниче поняття «Автоматика», а саме автоматизація технологічних машин галузей легкої промисловості і верстатів машинобудування для виготовлення таких машин та автоматизація технологічних процесів на засадах машин і верстатів. Автоматизація технологічних машин базується на професійне орієнтованих дисциплінах, наприклад, таких, як «Основи РКТМ», «Механічна технологія та обладнання», «Основи автоматики і автоматизації технологічних процесів та машин», «Схемотехнічне проектування машин» та інших. Автоматизація технологічних процесів, що відбуваються в апаратах або з допомогою апаратів базується на професійне орієнтованих дисциплінах «Процеси і апарати» та інших. Фундаментальними дисциплінами для автоматизації машин и для автоматизації технологічних процесів є математика, фізика, хімія, теоретична механіка, опір матеріалів, теорія механізмів і машин, гідравліка, гідро- та пневмопривод, електротехніка, електроніка та мікропроцесорна техніка та інші технічні дисципліни, які вивчаються за навчальними планами галузі знань «Механічна інженерія» за спеціальністю «Галузеве машинобудування» та спеціальністю «Прикладна механіка» бакалаврського ступеня вищої освіти та відповідними «Освітними програмами».

Мехатроніка тісно пов'язана з ***CALS-технологіями***.

CALS-технології (**C**ontinuous **A**cquisition and **L**ife cycle **S**upport) – безперервна інформаційна (комп'ютерна) підтримка життєвого циклу виробів цільового призначення. *CALS-технології* складаються з ***CAD/CAM/CAE*** підсистеми безперервного комп'ютерного циклу розробки, створення і виготовлення нових виробів у різних галузях промисловості. Поширена скорочена назва україномовного поняття CALS – **ІПП** (**І**нформаційна **П**ідтримка **П**роцесів життєвого циклу виробів).

Розвиток *CALS-технологій* пов'язаний з появою, так званих, віртуальних виробництв, в яких процес створення специфікацій з інформацією для програмно керованого технологічного устаткування може бути розподілений в часі і просторі між багатьма організаційно автономними розробниками. Досягненнями CALS-технологій є легкість розповсюдження передових проектних рішень, можливість багатократного відтворення частин проекту в нових розробках та інші.

Сучасні технологічні машини легкої промисловості є механіко-технологічними системами цільового функціонального призначення, які складаються з механічного модуля, електро-механічного модуля (привода), модуля керування і модуля живлення. Мехатронний модуль може застосовуватися як окремий об'єкт з індивідуальним робочим органом (ножем, важелем, повзуном) та індивідуальним електро-, пневмо-, або гідроприводом і гнучкою системою керування. В технологічних машинах мехатронний модуль кінематичне не з'єднаний з головним валом, а з'єднаний з ним за допомогою датчику кута повороту і датчика частоти обертання валу. Виконавчий елемент мехатронного модуля (серводвигун, кроковий двигун, пневмоциліндр, гідроциліндр) автоматично переміщує за програмним циклом ведені ланки механізмів цільового призначення. Наприклад, в автоматизованих швейних машинах такими програмно керованими цикловими механізмами є механізм зупинення головного валу з голкою у заданому положенні, механізм притискної лапки, механізм реверсу зубчастої рейки, механізм обрізки ниток, механізм двокоординатних переміщень бордюрної рами (п'ялець) з матеріалом або фурнітуротримача, механізм затискання кінця обрізаної нитки та інші механізми, які раніше включалися/вимикалися оператором вручну або виконувалися з використанням жорстких багатокрокових кулачків-програмоносіїв. У автоматизованих в'язальних машинах до програмно керованих циклових механізмів віднесені механізми в'язання, механізми зсуву ушкових гребінок основ'язальних машин, механізми зсуву голочниць (фонтур) плоско в'язальних машин, механізми відбору/додавання голок, механізми зміни нитководів, механізми кулірних клинів для зміни довжини петель, механізми відтягування трикотажного полотна або деталей верхнього трикотажу при в'язанні та інші.

Також при підготовки інженерів-механіків (бакалаврів і магістрів) за спеціальностями «Галузеве машинобудування», «Прикладна механіка» (бакалаврський рівень вищої освіти) та студентів спеціальності «Галузеве

машинобудування», які навчаються за освітньою програмою «Обладнання легкої промисловості та побутового обслуговування» при вивченні програмного матеріалу дисципліни «CAM-технології комп'ютерно-інтегрованого обладнання» (магістерський рівень вищої освіти) доцільно вивчення типових закономірностей побудови мехатронних і роботехнічних систем на засадах бюджетних і малопотужних драйверів для двигунів малої потужності та поширених на практиці контролерів **Arduino** і мови програмування **Arduino-C** для **DIY**-гаджетів. DIY (вимовляється «Ді-Ай-Вай») це поширена англійська абревіатура, яка є скороченням від «**D**o **I**t **Y**ourself», тобто «Зроби це сам».

1.4. **Умовні позначення і конструктивні особливості типових елементів схем мехатроніки**

Знання умовних графічних міжнародних позначень елементів схем мехатроніки пов'язане з обов'язковим засвоєнням умовних міжнародних позначень елементів кінематичних принципових схем (схеми типу К3), елементів схем електричних принципових (схеми типу Е3), елементів схем пневматичних принципових (схеми типу П3), елементів схем гідравлічних принципових (схеми типу Г3) та елементів схем комбінованих принципових (схеми типу С3). Запам'ятовування умовних графічних позначень елементів таких схем є частковою альтернативою вивчення іноземних мов. На мові умовних позначень елементів технічних систем можна спілкуватися з любим фахівцем не володіючи досконало його мовою. Ця думка автора має підтвердження при вивченні графічних позначень опцій примітивів об'єктів-елементів і об'єктів-процесів сучасних систем 3D-моделювання, заснованих на сучасної парадигмі (*від латинської мови* paradigma — модель) об'єктно-орієнованого програмування (**OOP** – **O**bject-**O**riented **P**rogramming) і її нащадка (спадкоємця) - модельне-орієнованого проектування (**MBD** - **M**odel-**B**ased **D**esign).

Умовні позначення елементів мехатроніки на схемах регламентовано наступними Державними стандартами (ГОСТами):

1. ГОСТ 2.701-84. Схемы. Виды и типы. Общие требования к выполнению.

2. ГОСТ 2.781-96 ЕСКД. Обозначения условные графические. Аппараты гидравлические и пневматические, устройства управления и приборы контрольно-измерительные.

3. ГОСТ 2.702-75 ЕСКД. Правила выполнения электрических схем.

На рис. 1.5…1.26 і в таблиці 1.1 наведені деякі умовні позначення за Державними стандартами поширених типових елементів і функціональних модулів мехатронних систем.

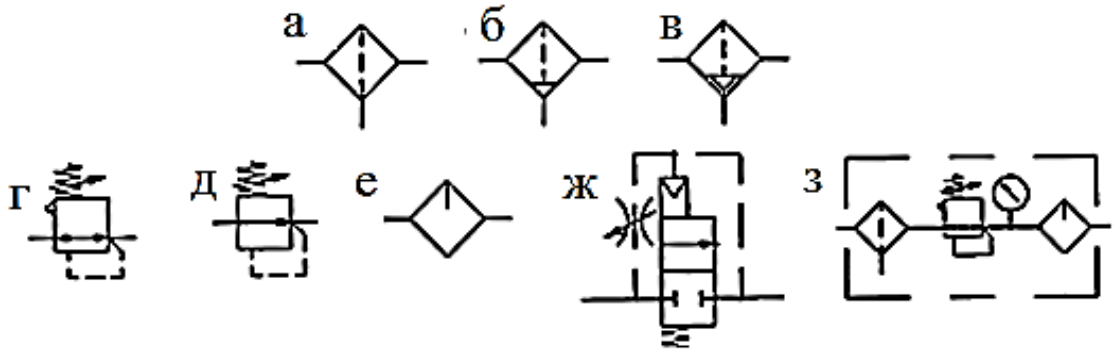


Рис. 1.5. Умовні позначення елементів підготовки повітря: а – фільтр; б – фільтр з ручним скиданням конденсату; в – фільтр з автоматичним скиданням конденсату; г – регулятор тиску зі скиданням надлишкового тиску; д – мастило розпилювач; е –мастило розпилювач; ж – клапан м'якого пуску з 50% тиску при включенні пневмо магістралі з наступним збільшення тиску стрибком до 100% тиску; з – модуль підготовки повітря (фільтр, регулятор тиску, манометр, мастило розпилювач)

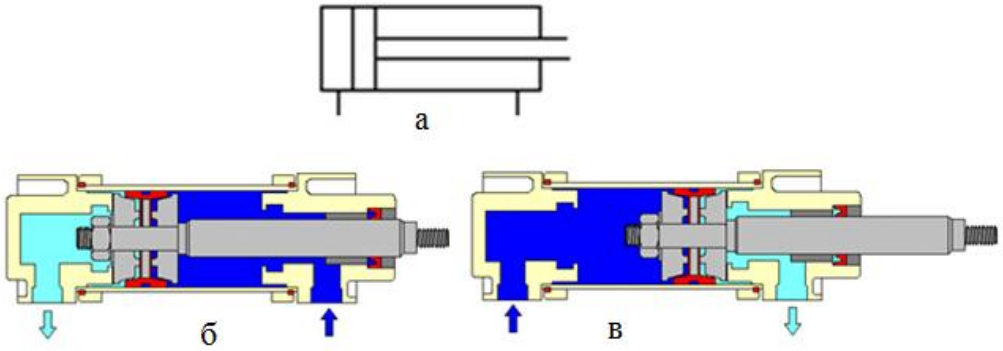


Рис. 1.6. Пневматичний немагнітний циліндр двосторонньої дії без демпфування поршня у кінці ходу: а – умовне позначення на схемах типу П3; б – конструктивна схема для положення поршню при подачі тиску в штокову камеру; в – конструктивна схема для положення поршню при подачі тиску у без штокову камеру циліндру

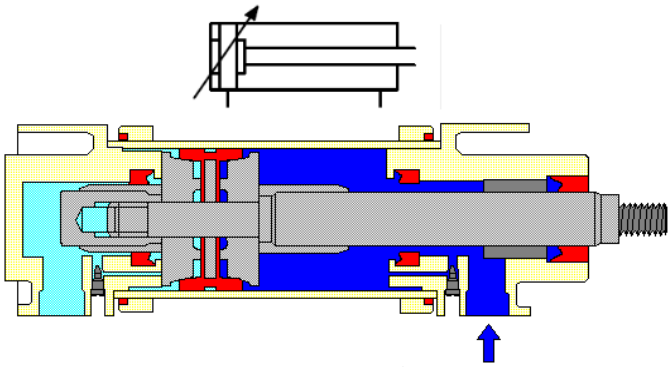


Рис. 1.7. Умовне позначення і конструктивна схема для пневматичного немагнітного циліндру двосторонньої дії з регульованим гальмуванням у кінці ходу

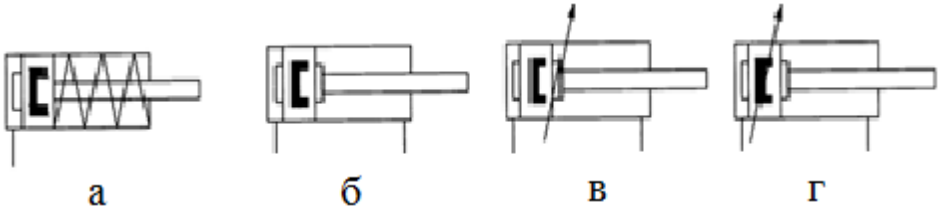


Рис. 1.8. Умовні позначення пневматичних циліндрів двосторонньої дії з магнітним диском поршню: а – зі зворотною пружиною; б – двосторонньої дії з нерегульованим гальмуванням у кінці ходу; в – двосторонньої дії з регульованим гальмуванням при висуванні; г – двосторонньої дії з регульованим гальмуванням в обох напрямах

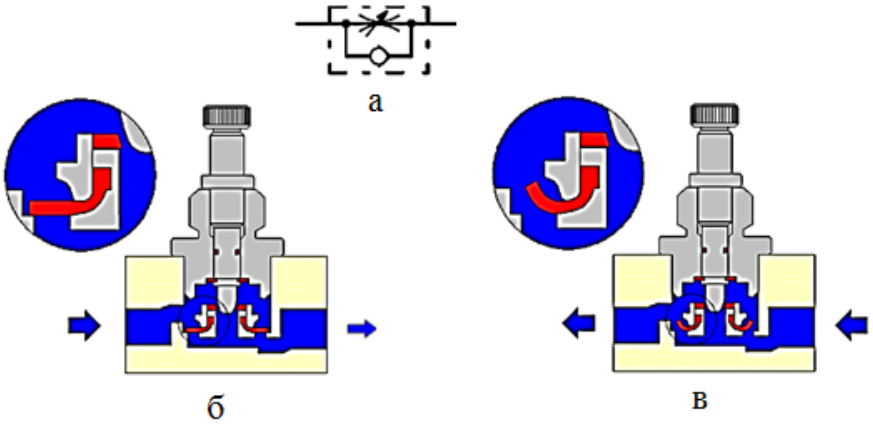


Рис. 1.9. Дросель зі зворотнім клапаном: а – умовне позначення на схемах типу П3; б – конструктивна схема для стану при регулювання витрати повітря у прямому напрямку; в – конструктивна схема для стану при вільному пропусканні повітря у зворотному напрямку

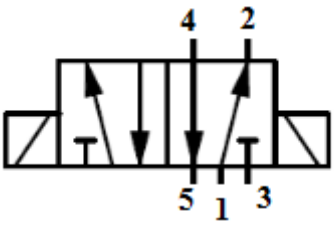
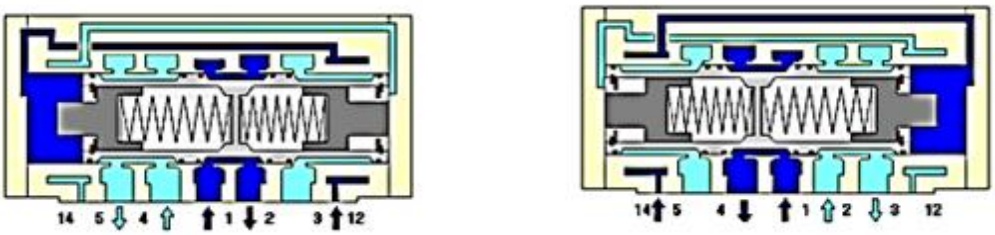


Рис. 1.10. Умовне позначення на схемах типу П3 пневмоозподільника п’яти лінійного двопозиційний (тип 5/2) з електромагнітним БІстабільним керуванням

а

б



в

Рис. 1.11. Умовні позначення пневморозподільників п’яти лінійних трипозиційних (тип 5/2) з пневматичним керуванням: а – БІстабільного; б – МОНОстабільного; в – конструктивні схеми для двох положень золотника при пневматичному БІстабільному керуванні

Рис. 1.12. Умовне позначення пневморозподільника п’яти лінійного двопозиційного (тип 5/2) з електромагнітним МОНОстабільним керуванням

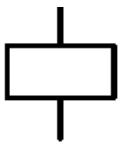


Рис. 1.13. Умовне позначення котушки електромагнітного реле на електричних схемах типу Е3 і комбінованих схемах типу С3

Рис. 1.14. Умовне позначення нормально розімкненого контакту реле і геркону (**гер**метичного **кон**такту) на схемах типу П3, типу Е3 і типу С3

Рис. 1.15. Умовне позначення нормально замкненого контакту електромагнітного реле

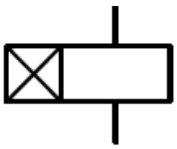


Рис. 1.16. Умовне позначення реле часу із затримкою спрацювання контакту на замикання (із затримкою по передньому фронту) – контакти реле спрацюють тільки після того як на реле часу буде подаватися струм протягом заданого часу і розімкнуться одразу після припинення подачі струму на котушку реле

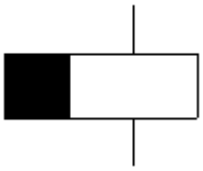


Рис. 1.17. Умовне позначення реле часу із затримкою спрацювання контакту на розмикання (із затримкою по задньому фронту) – контакти реле замкнуться одразу після подачі струму на реле, а після припинення подачі струму реле контакти реле будуть замкненими протягом заданого часу, після чого розімкнуться

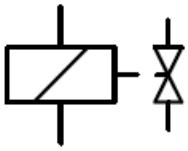


Рис. 1.18. Умовне позначення електромагніту (соленоїду) для переміщення золотника (міні-поршня) пневматичного розподільника

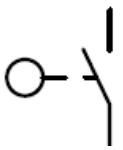
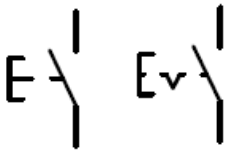
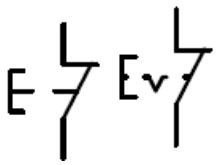


Рис. 1.19. Умовне позначення датчика положення (кінцевого вимикача) з роликом чи штовхачем з нормально розімкненим контактом



а б

Рис. 1.20. Умовне позначення кнопки з нормально розімкненим контактом **без фіксації** положення після натискання (а) і **з фіксацією** (б) положення у натисному стані



а б

Рис. 1.21. Умовне позначення кнопки з нормально замкненим контактом **без фіксації** положення (а) і **з фіксацією** положення (б)

Рис. 1.22. Умовне позначення пневматичної магістралі високого тиску

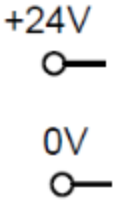


Рис. 1.23. Умовне позначення джерела жив-лення постійним струмом на схемах типу Е3

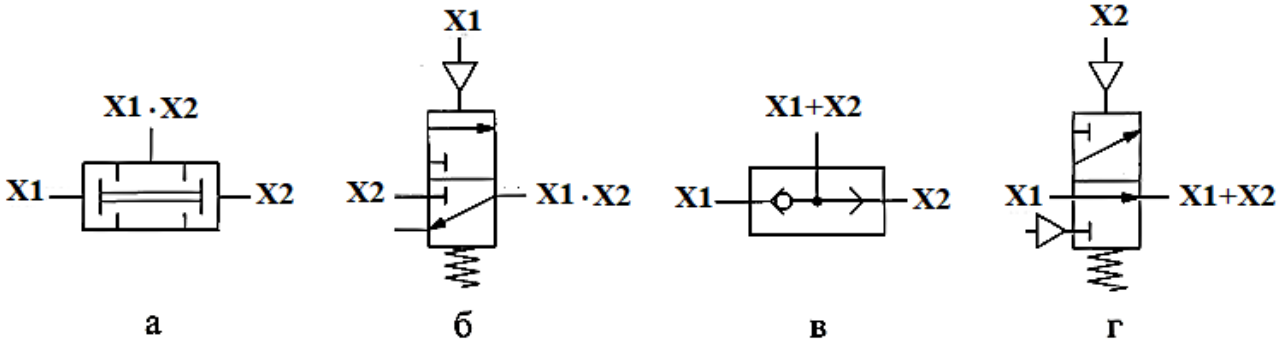


Рис. 1.24. Умовні позначення логічних елементів пневматичних схем (тип П3) мехатроніки: а – клапан для логічної операції "**І**" (AND – логічна операція множення або кон'юнкція); б – нормально закритий МОНОстабільний клапан для виконання логічної операції "**І**"; в – клапан для логічної операції "**АБО**" (OR – логічна операція складання або диз'юнкція); г – нормально відритий МОНОстабільний клапан для виконання логічної операції "**АБО**".

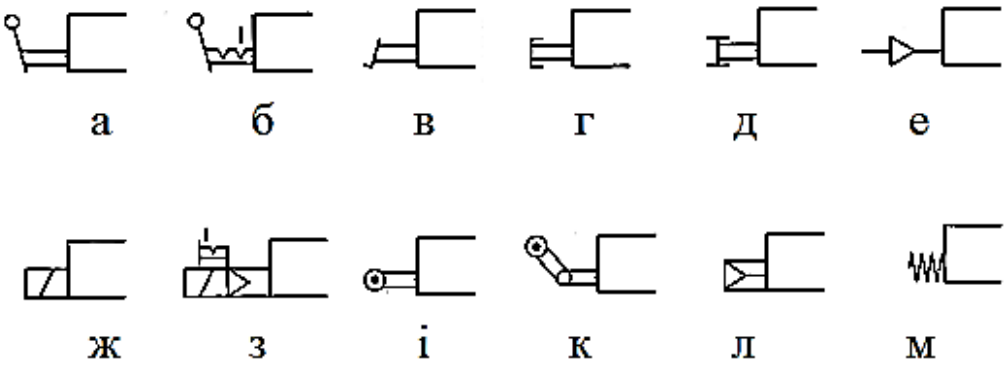
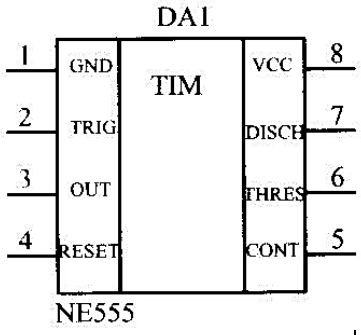
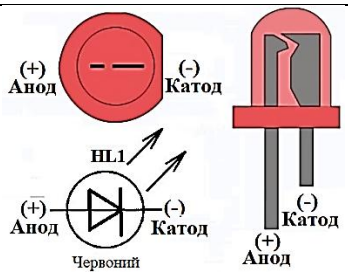
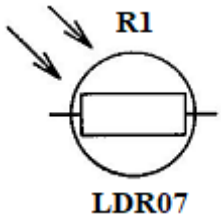
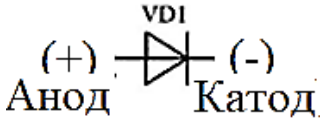
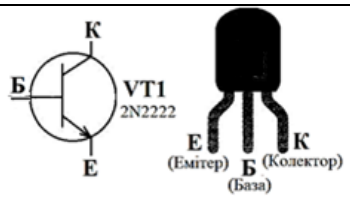
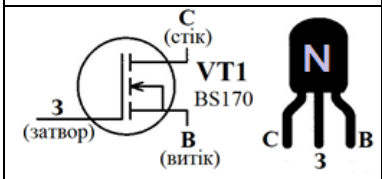


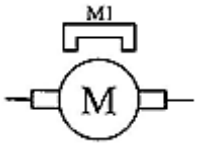
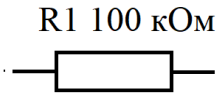
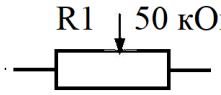
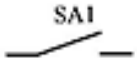
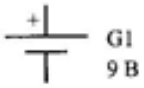
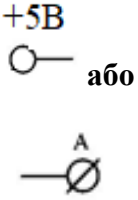
Рис. 1.25. Умовні позначення елементів керування і елементів контролю в схемах типу ПЗ: а – тумблер без фіксації положення; б – тумблер з фіксацією положення; в – педаль; г – кнопка; д – кнопка с фіксацією; е – пневматичне керування; ж – електромагнітне керування; з – електропневматичне керування з ручним дублюванням включення; і – механічне керування з роликовим штовхачем; к – механічне керування з роликовим штовхачем-важелем; л – повернення за допомогою пневматичної пружини; м – повернення за допомогою механічної пружини.

Рис. 1.26. Конструктивні схеми і електрична схема включення електромагнітного реле: а – стан реле, коли кнопка пуск S1 не натиснена, при цьому індикатори L1 та L2 відключені від джерела струми; б – стан реле, коли натиснута кнопка пуск S1, при цьому індикатор L1 включений, індикатор L2 – вимкнений.

В таблиці 1.1 наведені деякі умовні позначення деяких елементів електричних схем мехатроніки.

Таблиця 1.1. Умовні позначення елементів електричних схем мехатроніки

Графічне позначення	Назва	Текстове позначення
	Аналогова мікросхема таймера NE555	DA1– порядковий номер та назва контактів; 1…8 – номери контактів (виводи) мікросхеми
	Світлодіод(**LED**) світло-випро-мінюючий діод (**L**ight **E**mmiting **D**iode)	**HL1** – порядковий номер та напис коліру світла, який випромінює світлодіод, наприклад, червоний
	Фототранзистор	**R1**– порядковий номер; **LDR07** – тип фототранзистора
	Випрямний діод	**VD1**– порядковий номер та позначення на схемі
	Біполярний транзистор типу NPN	**VT1**– порядковий номер; **2N2222** – модель транзистора
	Польовий N-канальний транзистор	**VT1**– порядковий номер; **BS170** – модель транзистора; літера N позначає, що транзистор N-канальний

	Мотор (двигун) Постійного струму	**M1, M**– позначення на схемі
	Пьезодинамік	**BA1**– порядковий номер та позначення: «+» позитивний вивід, «-» непозначений вивід
	Конденсатор електролітичний	**C1** – позначення на схемі; 1 мкФ =$1 \cdot 10^{-6}$Ф ємкість конденсатора
	Конденсатор керамічний	**C1** – позначення на схемі; 1 мкФ =$1 \cdot 10^{-6}$Ф ємкість конденсатора
	Резистор	**R1** – порядковий номер; 100 кОм = $1 \cdot 10^{5}$ Ом – номінальний опір резистора
	Змінний резистор	**R1** – порядковий номер; 50 кОм = $5 \cdot 10^{4}$ Ом – максимальний опір резистора
	Кнопка з нормально відкритим контактом	SA1– порядковий номер в схемі
	Джерело живлення	**G1**- порядковий номер в схемі для напругі 9В; «+» - полярність
	Контакти електричних схем	Місце підключення та позначення джерела струму

1.5. **Типові функціональні і технологічні модулі мехатроніки**

Позначення типових функціональних модулів мехатроніки на схемах типу П3 наведені на рис. 1.27…рис.1.30. Якщо в схемі застосований БІстабільний розподільний пневмоклапан (пневморозподільник), то потрібно вмикати Y1 (пряма команда) і вимикати YN1(зворотна команда) або вмикати YN1 і вимикати Y1, тобто потрібно керувати двома електромагнітами (рис. 1.27), а якщо застосований МОНОстабільний розподільний пневмоклапан, то вмикати і вимикати потрібно один електромагніт Y2 (рис. 1.28).

Поширена скорочена назва типових функціональних модулів – електропривод, пневмопривод або гідропривод потребує наступного уточнення, яке розглянемо на прикладі визначення пневмопривода. *Пневмопривод* це сукупність трьох складових: компресора з електроприводом, типового *функціонального* пневматичного *модуля*, засобів комутації і захисту електродвигуна компресора.

Типовий *функціональний* пневматичний *модуль* це сукупність також трьох складових: пневмоциліндру (виконавчий механізм); пневморозподільника (передаточний механізм); циклової системи керування (релейної або за допомогою програмованого логічного контролера). При цьому системи керування виконавчим механізмом (пневмоциліндром) функціонального модуля можуть бути:

система керування *по положенню* штока або поршня;

система керування *по тиску* в робочої порожнині циліндру;

система керування *по часу* вистою поршня (штока) в крайніх положеннях.

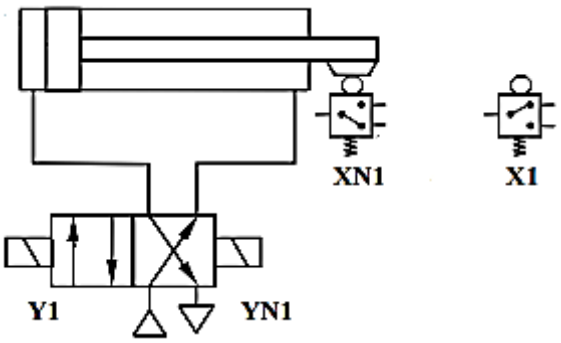


Рис. 1.27. Умовне позначення типового функціонального модуля на схемах типу П3 мехатронної системи на базі пневмоциліндра та БІстабільного пневмоклапана типу 4/2 з електромагнітним керуванням

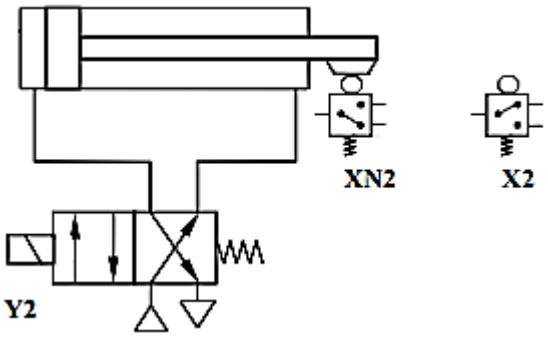


Рис. 1.28. Умовне позначення типового функціонального модуля на схемах типу С3 мехатронної системи на базі пневмоциліндра та МОНО-стабільного пневмоклапана типу 4/2 з електромагнітним керуванням

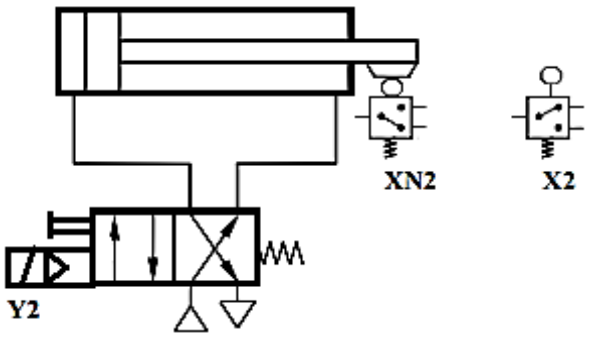


Рис. 1.29. Умовне позначення типового функціонального модуля на схемах типу С3 мехатронної системи на базі пневмоциліндра та МОНО-стабільного пілотного розподільника типу 4/2 з електропневматичним керуванням

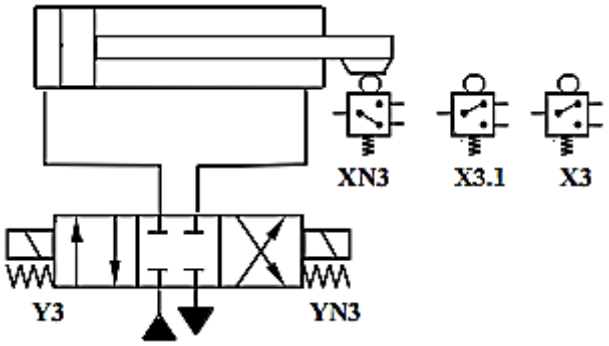


Рис. 1.30. Умовне позначення типового функціонального модуля на схемах типу С3 мехатронної системи на базі гідроциліндра та БІ-стабільного гідро розподільника типу 4/3 з електро магнітним керуванням

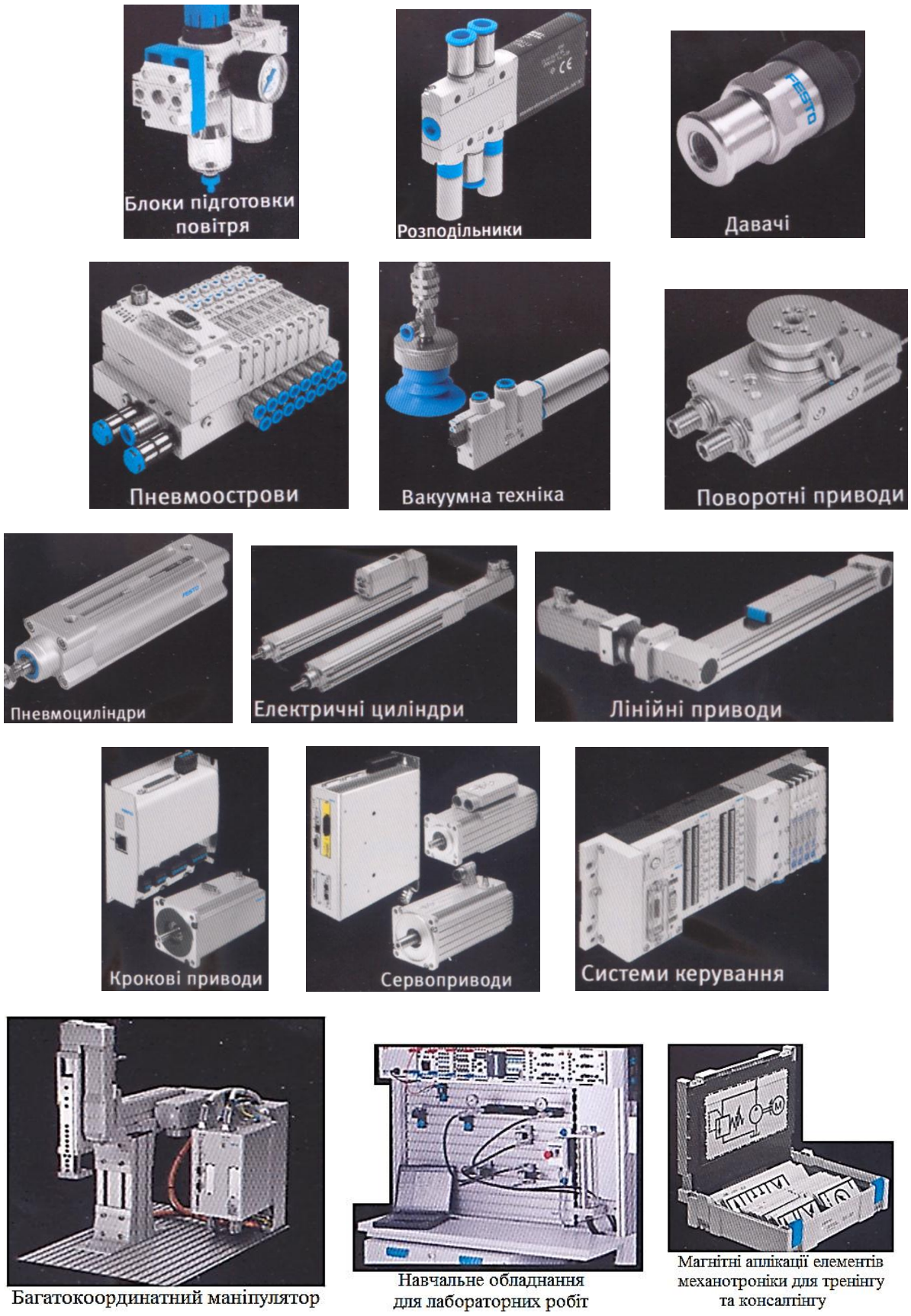


Рис. 1.31. Загальний вигляд деяких типових технічних і програмних засобів мехатроніки

На рис. 1.32 наведений загальний вигляд лабораторного стенду з типовими функціональними елементами мехатроніки для побудови мехатронних і роботехнічних систем типу CNC-верстатів машинобудування та технологічних CNC-машин легкої проомисловості.

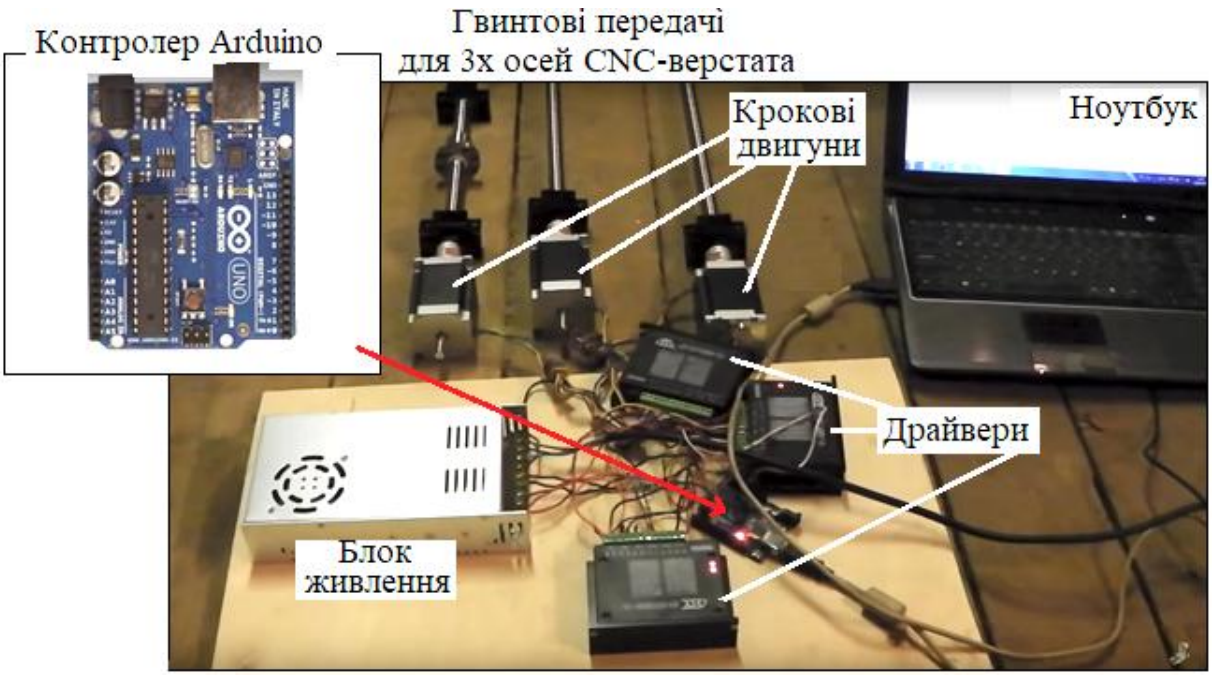


Рис. 1.32. Назва і загальний вигляд типових елементами мехатроніки для побудови мехатронних і роботехнічних систем

Персональний комп'ютер в мехатронної системі призначений:

для створення 3D або 2D CAM-моделей об'єктів-виробів, наприклад, в програмних середовищах ArtCAM, SolidCAM, CreoParametкшс (CAD/CAM) та інших для наступного їх перетворення постпроцесором CNC-верстата або CNC-машини технологічної. Постпроцесор врахувує в створенної CAM-моделі кінематику машин і верстатів з числовим програмним керуванням і перетворює CAM-модель у G-коди, M-коди та у інші коди запрограмованих траекторій і режимів руху робочих інструментів інструментів, наприклад, фрез, заготовок матеріалу, каретки, порталу CNC-верстатів, п'ялець або бордюрної рами вишивальних CNC-машин та інших технологічних машин, які реалізують цифрові технології;

для програмування циклів роботи мехатронних пристроїв та технічних систем;

для програмування режимів та дизайну виготовлення об’єктів з різних матеріалів.

Контролер в мехатронної системі виконується на мікроконтролері і тому його поширена назва «*мікроконтролер*». Контролер має спрощену структуру міні-ЕОМ і призначений для роботи з командамі, операндми і операторами програм, має порти входів/виходів для підключення датчиків, виконавчих механізмів та інших зовнішніх пристроїв. Контролер (плата контролера) за допомогою *пінів* і *клем* (цифрових, аналогових, живлення) стикується з драйвером/драйверами і блоком живлення, за допомого USB-кабеля – з комп’ютером, а з допомогою шилда – з іншим електронним модулем цільового призначення. В пневматичних і комбинованих схемах мехатроніки окрім дротів використовують пластикові *пневмотрубки*, з’єднення яких з типовими функціональними модулями на рис. 1.27…рис. 1.29 виконуються за допомого *фітінгів* різних конструкцій, а на рис. 1.30 – за допомогою *гідрошлангів* і *штуцерів*.

Виконавчі механізми обертової дії у вигляді *крокового приводу* або *сервоприводу* в мехатронної системі призначені для лінійних, зворотно-поступових, обертових або коливних переміщень веденої ланки важільних механізмів в технологічних машинах і верстатах. Виконавчі механізми зворотно-поступової дії у вигляді гідро- і пневморозподільників призначені для вмикання/вимикання гідро- пневмоциліндрів цільового призначення. Виконавчі механізми зворотно-поступової дії у вигляді електромагнитів і реле призначені для вмикання/вимикання нагрівачів, насосів, клапанів та інших пристроїв при автоматичному регулюванні температури, рівня рідини, концентрації, в’язкості та інших неелектричних фізічних параметрів у ***технологічних апаратах*** цільового призначення.

Драйвер. Класичне визначення драйверу – це програма, що управляє роботою контролера. Поширена назва «*драйвер»* в мехатронної системі це також електронний пристрій для сполучення контролера з виконавчими механізмамі, датчикамі, цифровими дісплеями пультів керування та іншіми електромеханічнимі та електронними пристроями мехатроніки та робототехнічних систем.

Блоки живлення в мехатронної системі призначені для електроживлення різних споживачів енергії. Джерела живлення поділяються на первинні і вторинні. До першої групи відносяться перетворювачі. В мехатроніки для живлення драйверів застосовують блоки живлення постійного струму на потужність до 600 Ватт і більше в

залежності від цільового призначення виконавчих механізмів. Для роботи логіки драйвера на рис.1.32 потрібна напруга +5 V або +3,3 V від блоку електроживлення комп'ютера та окремий блок живлення +48V, 12A для крокових двигунів, наприклад, поширених крокових двигунів типу NEMA 17, NEMA 23 та ін.

В таблиці 1.2 наведені умовні позначення (символи) автоматизованих операцій виробничих процесів на функціональних схемах мехатроніки за вимогами стандарту **VDI 2860** (Німеччина), а в таблиці 1.3 – умовні позначення і стан сигналів логічних елементів блок-схем мехатроніки.

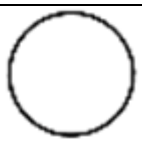
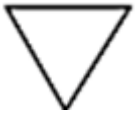
VDI 2860 – це стандарт (**V**erband **D**eutscher **I**ngenieure – Асоціація німецьких інженерів), який включає умовні позначення наступних основних і допоміжних технологічних операцій циклу робочого процесу у функціональних схемах мехатроніки:

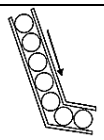
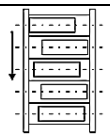
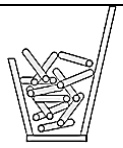
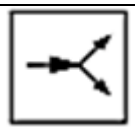
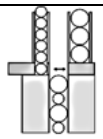
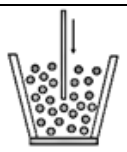
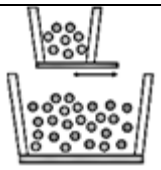
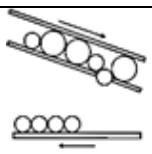
основні символи для позначення **обробки, виготовлення та перевірки**;

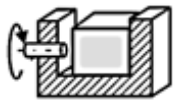
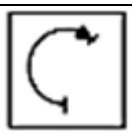
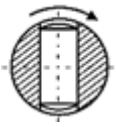
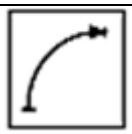
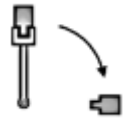
символів елементарних функцій при виконанні операцій **розділення, комбінування, повороту, переміщення, утримання, випуску, тестування**;

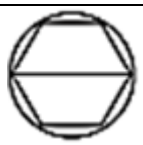
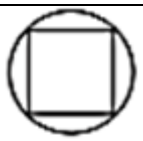
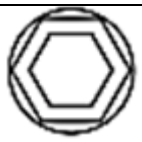
символи додаткових функцій для позначення різних зберігання (**магазин, бункер, штабелер**) та операцій внутрвпроцесної послідовної або паралельної передачі заготовки, деталі або фурнітури.

Таблиця 1.2. Умовні позначення технологічних модулів мехатроніки

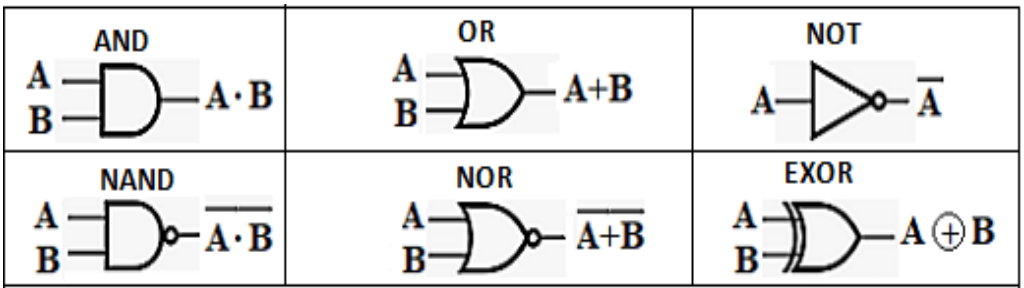
Умовні позначення	Функції або призначення операцій в цикле	Приклад застосування
	виробничий процес (основний символ)	
	обробка (основний символ)	
	Перевірка, контроль (основний символ)	

	впорядковане зберігання (магазин)	
	частково впорядковане зберігання	
	безладне зберігання (бункер)	
	роз’єднання потоку деталей	
	об’єднання потоку деталей	
	перегороджування сипучого матеріалу	
	додавання сипучого матеріалу	
	розподіл по n-деталей	
	розподіл по n-деталей і переміщувати на наступну позицію	
	сортирувати деталей по розміру	

	затискання деталі	
	розблокування деталі	
	зберігання / зберігання без дії сили	
	поворот деталі	
	перенесення деталі з її поворотом	
	рух, переміщення,	
	позиціонування	
	сортування деталей, контроль орієнтування деталі	
	пересилання тому щоб подати далі	
	пересилання з одночасною підтримкою орієнтації на заготівлю	

	операція тестування	
	операції зміни форми (формування, різання)	
	операція приєднання (складання)	
	креативне формування (3D-друк)	
	операції покриття, модифікація властивостей матеріалу	

Таблиця 1.3. Умовні позначення і стан сигналів логічних елементів блок-схем мехатроніки



Input	Output	Input		Output				
		A	B	AND	NAND	OR	NOR	EXOR
NOT		0	0	0	1	0	1	0
A	$\overline{A}$	0	1	0	1	1	0	1
0	1	1	0	0	1	1	0	1
1	0	1	1	1	0	1	0	0

Контрольні питання до розділу 1:

1.Які зв’язки мехатроніки з **C**omputer-**A**ided **D**esign ?

2. Які зв’язки мехатроніки з **C**omputer-**A**ided **M**anagement ?

3. Які зв’язки мехатроніки з **C**omputer **N**umerical **C**ontrol ?

4. Які зв’язки між CAD і CAM системами ?

5. Чим відрізняється технологічна машина від механіко-технологічної системи з мікропроцесорним керування ?

6. Що означає «Programmable Logic Controller» ?

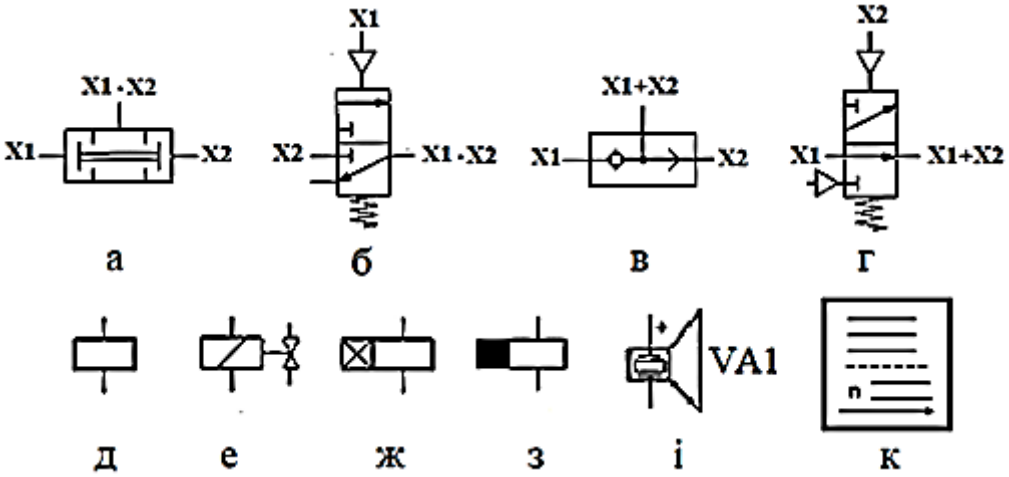
7. Які технічні засоби використовують в мехатрониці?

8. Які програмні засоби використовують в мехатрониці?

9. З яких слів утворена назва «геркон» і чим він відрізняється від кінцевих вимикачів на рис.1.27…1.30 ?

10.Яке призначення персонального комп’ютера, контролера, крокових двигунів, механічних передач, драйвера та блоків живлення в мехатронних і робототехнічних системах, пристроях CNC-машинах і CNC-верстатах ?

11.Що позначають наступні умовні позначення елементів мехатроніки і у яких видах і типах схем використуються такі позначення ?



2. СИСТЕМИ КЕРУВАННЯ ТЕХНОЛОГІЧНИМИ МАШИНАМИ ЛЕГКОЇ ПРОМИСЛОВОСТІ І ВЕРСТАТАМИ МАШИНОБУДУВАННЯ

2.1.**Класифікація систем керування технологічними машинами і верстатами**

Розрізняють жорсткі і гнучкі системи керування технологічними машинами і верстатами.

1. ***Жорсткі системи керування машинами:***

1.1. Система керування типу «розподільний вал»;

1.2. Система керування з програмоносієм типу «багатокроковий кулачок» машин легкої промисловості.

2. ***Гнучкі системи керування машинами*** - це системи керування з контролером та можливостями комп'ютерного програмування і перепрограмування циклу роботи. Найбільше застосування в автоматизованих машинах легкої промисловості отримали наступні Гнучкі системи керування;

2.1.***Числове програмне керування***. Поширена назва - система керування типу **CNC** (**C**omputer **N**umerical **C**ontrol*)*, а машини і верстати з такими системами керування мають поширену назву CNC-машини і CNC-верстати;

2.2.***Циклове програмне керування*** (***ЦПК***).

Для систем автоматизованого керування технологічними машинами галузей легкої промисловості і верстатами галузі машинобудування застосовується наступна класифікація.

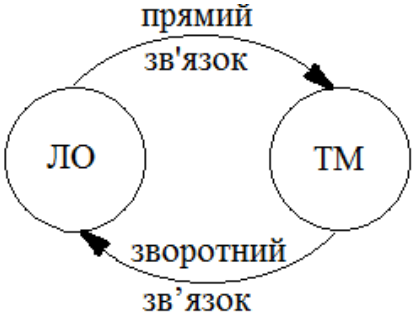


Рис. 2.1. Схема до класифікації основних понятий взаємодій у системі: «*людина оператор* (ЛО) - *технологічна машина* (ТМ)»

Якщо автоматизований прямий зв'язок то це ***автоматизоване керування*** або система автоматизованого керування. Якщо автоматизований зворотній зв'язок то це ***автоматизований контроль*** або автоматизована сигналізація. Якщо автоматизовані і прямий зв'язок і зворотній зв'язок то це ***автоматизоване регулювання*** або система автоматизованого регулювання.

Дамо наступні доповнення до наведених основних визначень.

Автоматичні системи стабілізації - забезпечують підтримання регульованої фізичної величини на заданому рівні з потрібною точністю. До таких систем відносяться, наприклад, система регулювання температури в пресах ВТО швейних виробів і сушильних установках для

взуття, система регулювання швидкості двигуна та багато інших. Системи стабілізації поділяються на статичні та астатичні.

Статичними системами називаються такі, в яких регульований параметр фізичної величини в сталому режимі змінюється при зміненнях збурюючої дії.

Астатичні системи забезпечують підтримку регульованого параметру фізичної величини в сталому режимі залишається незмінним при зміненнях збурюючої дії. Тобто, *астатичні системи* забезпечують регулювання змінних в сталому режимі без похибки, а саме підтримують його на заданому рівні, а *статичні системи* – з деяким відхиленням (похибкою).

Слідкуючі системи - здійснюють відпрацювання завдання регульованої величини в часі за законом, якій заздалегідь або невідомий, або відомий тільки в миттєвий час. Вхідний вплив визначається тільки в процесі функціонування системи. Прикладами такої системи можуть служити технологічні машини для з'єднання деталей по контуру, у яких контур зрізу деталей з текстилю або шкіри відстежується фотоелектричними сенсорами послідовно в часі тільки при переміщенні деталей під притискною лапкою машини.

Системи програмного керування - здійснюють зміну регульованих величин в часі за законами, які заздалегідь запрограмовані. До таких систем відносяться, зокрема, системи CNC або CNC-верстати (верстати з CNC).

Адаптивні системи - здійснюють оптимальне, за заданим показником якості, керування поточним станом об’єкта при зміненнях умов його роботи. До класу адаптивних відносяться системи, що само настроюються, самоорганізуються, самонавчаються, наприклад у робототехнічних системах.

Система керування *промисловим роботом* є багаторівневою. Її нижньому рівню у вигляді координатних приводів відводиться особлива роль, оскільки саме на цьому рівні вирішуються задачі забезпечення

потрібної якості відпрацювання рухів, що задаються. Ці задачі ускладняються тим, що процеси функціонування приводів роботів мають наступні специфічні особливості:

по-перше, статична і динамічна складові навантаження двигуна можуть сильно змінюватись у зв'язку зі зміною положень ланок розімкнутої кінематичної схеми робота;

по-друге, можливі режими роботи двигуна в загальмованому стані при роботі «на упор», що приводить до нагріву кінематичних пар;

по-третє, до електроприводів роботів пред'являються, як правило, найбільш високі вимоги, а саме: необхідність забезпечення астатичного режиму (без пере регулювання) відпрацювання заданих переміщень схвату для виключення ударів по обладнанню, що обслуговується; забезпечення широкого діапазону регулювання по швидкості та позиціонуванню.

Як було позначено, програмне керування технологічними машинами легкої промисловості і верстатами машинобудування поділяють на ***циклове програмне керування*** (ЦПК) і ***числове програмне керування*** (CNC).

Об'єкти з ЦПК потребують використання програм і контролера, які містять інформацію про цикл і режими обробки об'єкту з регульованими приводами головного руху і подачі від шляхових перемикачів. На рис. 2.2 наведена блок-схема інтерфейсу програмуючого контролера комп'ютерно-інтегрованої швейної машині.

Найпростішим пристроєм ЦПК є кулачкові командо-апарати, які видають команди на шляхові (кінцеві) перемикачі, що забезпечують початок руху або припинення руху відповідних робочих органів технологічної машини або верстату. Предметом проектування в дисципліні «Мехатроніка в галузевому машинобудуванні» є електричні, пневматичні, гідравлічні або комбіновані – електропневматичні/електрогідравлічні системи ЦПК технологічних машин легкої промисловості і верстатів машинобудування.

Об'єкти з *числовим програмним керуванням* (CNC-машини) потребують використання програм, що містять інформацію про креслення деталі, про цикл і режими обробки, про переміщення заготовки та інформацію про інструмент, записану у вигляді певної послідовності

чисел, що представляє собою мову програмування. У системах CNC на всьому шляху підготовки програми керування аж до її передачі робочим органам верстата маємо справу тільки з інформацією в цифровій формі.

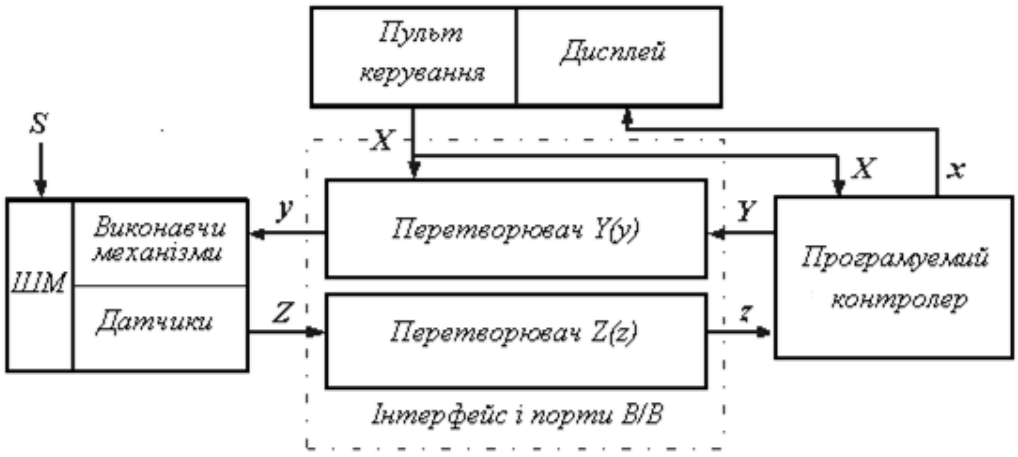


Рис. 2.2. Інтерфейс програмуючого контролера в блок-схемі комп'ютерно-інтегрованої швейної машині з цифровим керуванням

CNC-машини за ознаками переміщення робочих органів можна класифікувати на:

- позиційні системи керування;
- контурні системи керування;
- комбіновані системи керування;
- універсальні системи керування;

Системи CNC за ознаками наявності зворотного можна класифікувати на:

- розімкнуті системи керування;
- замкнуті системи керування.

В ***позиційних CNC-системах*** обробка здійснюється в процесі почергового або одночасного переміщення робочих органів верстата за осями координат OX, OY, OZ.

В ***контурних CNC-системах*** обробка забезпечується переміщенням робочих органів верстата по траєкторії (зазвичай криволінійної) і з контурній швидкістю, заданими програмою. Контурна швидкість є результуюча швидкість подачі робочих органів верстата, напрямок якої збігається з дотичною в кожній точці заданого контуру обробки.

До CNC-машин відносяться вишивальні автомати і швейно-вишивальні машини, розкрійні агрегати, принтери, графопобудовники, дігитайзери, координатні верстати металообробки та інші.

Рис. 2.3. Класифікація програмоносіїв механіко-технологічних систем на прикладі технологічних машин легкої промисловості

До неавтоматизованих машин з ЦПК відносяться технологічні машини із жорсткою системою керування типу ***«розподільний вал»***. Вони мають механізми, що кінематичне не з'єднані з головним валом і роботу таких механізмів потрібно автоматизувати. Наприклад, в швейних машинах це механізм притискної лапки, який кінематичне з'єднаний з регулятором натягу голкової нитки, а не з головним валом машини, механізм реверсу (зворотного переміщення матеріалу при виготовленні закріпок по кінцям шва на машинах човникового стібка), механізм зсуву голочниці плосков'язальних машин, механізм зсуву гребінок основов'язальних машин, механізм переміщення клинів і платин в'язальних машин, функціональна група відбору трикотажних голок механізму в'язання тощо.

Технологічні CNC-машини мають поширену назву «***машини з електронним керуванням***» або «***машини з цифровим керуванням***». Технологічні машини із жорсткою системою керування і з гнучкою системою керування мають, відповідно, жорсткі і гнучкі програмоносії (рис. 2.3).

В мехатронних системах з ЦПК потрібно розрізняти БІстабільне керування і МОНОстабільне керування.

БІстабільне керування **-** це двостороннє програмне керування з використанням БІстабільних розподільників пневмоциліндром/ гідроциліндром. В алгоритмі БІстабільного керування у явному виді повинна бути *дві команди*: перша команда це *пряма команда* і друга команда це *зворотна команда*. Електромагніти БІстабільного пневморозподільника запам'ятають останню команду і залишається в тому же стані, в який його перевела пряма команда, наприклад SET=1 і RESET=0 до того часу, поки ні з'явиться зворотна команда RESET=1 і SET=0. До відповідних портів виходів «*Output*» контролера підключаються електромагніти Y1 і YN1 пневморозподільника, а до портів входів «*Input*» контролера підключаються датчики X1 і XN1 кінцевих положень штоку поршня пневмоциліндру та кнопки S_i стану циклу та інші датчики.

МОНОстабільне керування **-** це одностороннє програмне керування з використанням МОНОстабільних розподільників виконавчім механізмом, при якому в явному вигляді повинна бути одна основна команда, а не дві команди (пряма і зворотна), як при БІстабільному керуванні. Тому при любому зникненні основної команди відбувається повернення виконавчого механізму у початковий стан. Сигнал контролю положення виконавчого механізму підводиться до одного порту контролера. Тому, якщо в програмі сигнал X1=1 тоді виконавчий механізм включений, якщо X1=0 – вимкнений і на відповідному виході контролера електричний сигнал відсутній.

Проектування сучасних машин легкої промисловості, як складних механіко-технологічних систем з мікропроцесорним керуванням направлено на їх автоматизацією. Швейні, в'язальні та взуттєві машин є типовими представниками машин легкої промисловості. Але якщо швейні машини і циклові напівавтомати мають переважно жорстку систему керування типу «*розподільний вал*» (типу «РВ»), то взуттєві обтяжно-затяжні машини і вирубальні преси з гідроприводом – *циклову систему керування*. В'язальні машини мають і жорстку систему керування типу «РВ» (основов'язальні машини), і гнучку циклову систему керування, наприклад круглов'язальні і плосков'язальні машини, а також панчішні машини-автомати.

В машинах легкої промисловості застосовуються цільові механізми з ортогональною і паралельною кінематикою та з декількома індивідуальними приводами на засадах крокових або вентильних електроприводів, електромагнітних приводів та/або пнемо-/гідроприводів. В швейно-вишивальних машинах і у вишивальних автоматах машинах з *числовим програмним керуванням* використовуються важільні механізми з ортогональною кінематикою, до яких відносяться 2D-механізми переміщення матеріалу, а до механізмів з паралельною кінематикою відносяться інші механізми цих машин з індивідуальними приводами і цикловим програмним керуванням на засадах мехатроніки. Застосування мехатроніки в галузевому машинобудуванні легкої промисловості передбачає переведення технологічних машин з класу неавтоматизованих в клас автоматизованих технологічних машин.

Неавтоматизована *швейна машина човникового стібка* це *електромеханічна механіко-технологічна система* з ручним керуванням, призначенням якої є узгоджене за циклограмою роботи переплетення верхньої (голкової) і нижньої (човникової) ниток в пакеті матеріалів при утворенні стібків, яка має корпус (головку машини) та електропривод. При цьому корпус машини складений з рукава і платформи, в рукаві змонтований головний вал, механізм голки, механізм ниткопритягувача, притискна лапка і регулятор натягу голкової нитки, на платформі закріплена голкова пластина, а під платформою змонтований механізм човника і механізм зубчатої рейки з регулятором довжини стібка і реверсу матеріалу. До таких механіко-технологічних систем цільового призначення віднесена більшість неавтоматизованих електромеханічних *швейних машин*, поширена назва яких «*швейні машини човникового стібка» та «швейні машини ланцюгового стібка».* На базі таких машин будуються автоматизовані машини і напівавтомати з електронним (комп’ютерним) керуванням.

Неавтоматизована *швейна машина ланцюгового стібка* це *електромеханічна механіко-технологічна система* з ручним керуванням, призначенням якої є узгоджене за циклограмою роботи переплетення системи голкових (верхніх) ниток із системою ниток петельників (нижніх ниток) під матеріалом, над матеріалом та збоку зрізу матеріалу при утворенні стібків, яка має корпус (головку машини) та електропривод. При

цьому корпус машини складений з рукава і платформи, в рукаві змонтований голковий вал і механізм голки, ниткоподатчик, притискна лапка і регулятор натягу ниток, на платформі закріплена голкова пластина, а під платформою змонтований головний вал, механізм(и) петельника(ків), ниткоподатчик і механізм зубчатої рейки. На базі машин ланцюгового стібка будуються автоматизовані машини і напівавтомати з електронним (комп'ютерним) керуванням.

Якщо такі *електромеханічні механіко-технологічні системи* оснащені мехатронними функціональними модулями цільового призначення і системами керування з пам'яттю на принципі жорсткої архітектури і логіки, тобто ***оператор*** *не може втручатися в пам'ять машини* (оператор не може змінювати програму роботи машини), то неавтоматизовані електромеханічні швейні машини переходять в *клас автоматизованих швейних машин*. Наприклад, автоматизовані швейні машини загального і спеціального призначення зі стоп-мотором, в тому числі швейні напівавтомати для виготовлення петель, для обробки обтачних кишень в рамку, пришивання фурнітури, зшивання деталей по контуру та інші це електронні ***швейні машини із жорсткою логікою системи керування.***

Якщо *електромеханічні механіко-технологічні системи* оснащені вбудованим логічним контролером, що має архітектуру комп'ютера для керування електроприводом головного валу та індивідуальними приводами на електромагнітних, крокових або вентильних сервоприводах по сигналам датчиків цільових механізмів машини і при цьому *оператор може втручатися в пам'ять машини* для програмування і гнучкої зміни програм й алгоритму роботи машини, то електромеханічні швейні машини переходять в ***клас*** *комп'ютерно-інтегрованих швейних машин*. Наприклад, вишивальні машини-автомати, швейно-вишивальні машини та інші машини з числовим програмним керуванням – поширена назва CNC-машини, які є *комп'ютерно-інтегрованими швейними машинами*.

В'язальні машини є також механіко-технологічними системами, які призначені для виробництва трикотажного полотна, деталей та виробів в'язальним або кулірним способом з переплетених петель з текстильної пряжі за певною послідовністю операцій, які обумовлені механічної

технологією петле утворення та конструктивними особливостями механізмів ниткоподачі і зміни ниток, механізмів в'язання *трикотажними голками* на плоскої або циліндричної голочниці і механізму відтяжки виробу або напівфабрикату.

Якщо така механіко-технологічна система має електропривод то вона перетворюється в *електромеханічну механіко-технологічну систему*, а саме в *електромеханічну плосков'язальну машину-напівавтомат* або в *електромеханічну круглов'язальну машину-автомат* класу панчішно-шкарпетковий автомат.

Якщо *електромеханічна плоска в'язальна машина* оснащена електронними елементами вбудованого пристрою керуванням програмованим відбором голок (для встановлених рапортів переплетень і зміни кольору пряжі в малюнках), має пам'ять на принципі жорсткої архітектури і логіки, тобто *оператор не може змінювати пам'ять машини* поки не замініть на новий програмоносій-перфокарту малюнку, то таки машини переходять в клас *електронних плосков'язальних машин*. Наприклад, автоматизовані перфокарточні *плосков'язальні машини* моделі Veritas KM-245P, моделі К-747 фірми Тойота (Японія) з перфокартами з рапортом на 12 голок, перфокарточні *плосков'язальні машини* моделі модель Silver Reed SK-280 (Японія/Китай) та інші з перфокартами з рапортом на 24 голки є *електронними плосков'язальними машинами.*

Якщо *електромеханічна плоска в'язальна машина* оснащена вбудованим вільно програмуючим контролером для керування електромагнітними приводами на соленоїдах для відбору голок або на крокових сервоприводах, для програмованого повороту клинів механізму в'язання за програмою що задається з клавішного пульта та дисплеєм або програмування через підключення до персонального комп'ютера, тобто *оператор може втручатися в електронну пам'ять машини* для програмування і гнучкої зміни програми (алгоритму) роботи машини, то таки в'язальні машини переходять в клас *комп'ютерних плосков'язальних машин*. Наприклад, автоматизовані побутові плосков'язальні машини моделі *Brother KH-765i* (Японія), моделі *Silver Reed SK-860,* моделі *Passap Electronic 6000* (Швейцарія), моделі *Veritas*

КМ-245С та промислові плосков'язальні машини фірми *Shima Seiki* (Японія) і фірми *Stoll* (Німеччина) та інші є *комп'ютерно-інтегрованими плосков'язальними машинами.*

Всі в'язальні машини, в тому числі кулірні, однофонтурні і двофонтурні, плоскі і круглі, окрім основов'язальних машин мають механізми в'язання, які за прийнятою в ТММ термінологією є кулачковими механізмами зі зворотно-поступовим рухом штовхача. Кулачками є клини замкової системи, а штовхачами – трикотажні голки. Клини замкової системи в'язальної каретки, а ні каретка, як помилково наведено в деяких виданнях, забезпечують переміщення голок по законам, які реалізують основні 9…11 технологічних операцій петле утворення. На засадах механічної технології відбувається періодичний зворотно-поступовий рух трикотажних голок з частотою руху каретки вправо-вліво відносно голочниці. Основними елементами структури трикотажу є ***петлі, накиди*** і ***протяжки***, кількісне сполучення між якими утворює різні структури трикотажу (різні трикотажні переплетення) при взаємодії між собою системи ниткоподачи, механізму в'язання і системи відтягування трикотажного полотна або деталі трикотажного виробу.

Комп'ютерна в'язальна машина моделі КН-965і Brother (Японія) ***є типовою двофонтурною плосков'язальною машиною, у який закладені основні принципи будови сучасних мехатронних плосков'язальних машин.*** Функціональна схема комп'ютерної плосков'язальної машини моделі КН 965і Brother (Японія) наведена на рис.2.4. Машини *Brother* мають голочницю і в'язальну каретку, які кінематичне з'єднані між собою язичковими голками.

Запрограмований відбір голок виконується за допомогою 16 реле постійного струму (міні соленоїдів) по командах від вбудованого міні-комп'ютера (вільно програмованого мікропроцесорного контролера). Зворотно-поступальний рух в'язальної каретки виконується від електродвигуна з редуктором.

Функціональна схема комп'ютерної плосков'язальної машини наведена на рис. 2.4, де прийняті наступні позначення:

CPU – мікроконтролер;

ЕД – електродвигун приводу;

КЗ – каретка задня (каретка передня по відношенню до комп'ютера);

КП – каретка передня (каретка задня по відношенню до комп'ютера);

РЕМ – реле електромагнітні постійного струму (міні-соленоїди);
ПК – передача конічна;

«В»– робоча позиція (РП) голок;

«*D*» – передня робоча позиція («ПРП») голок

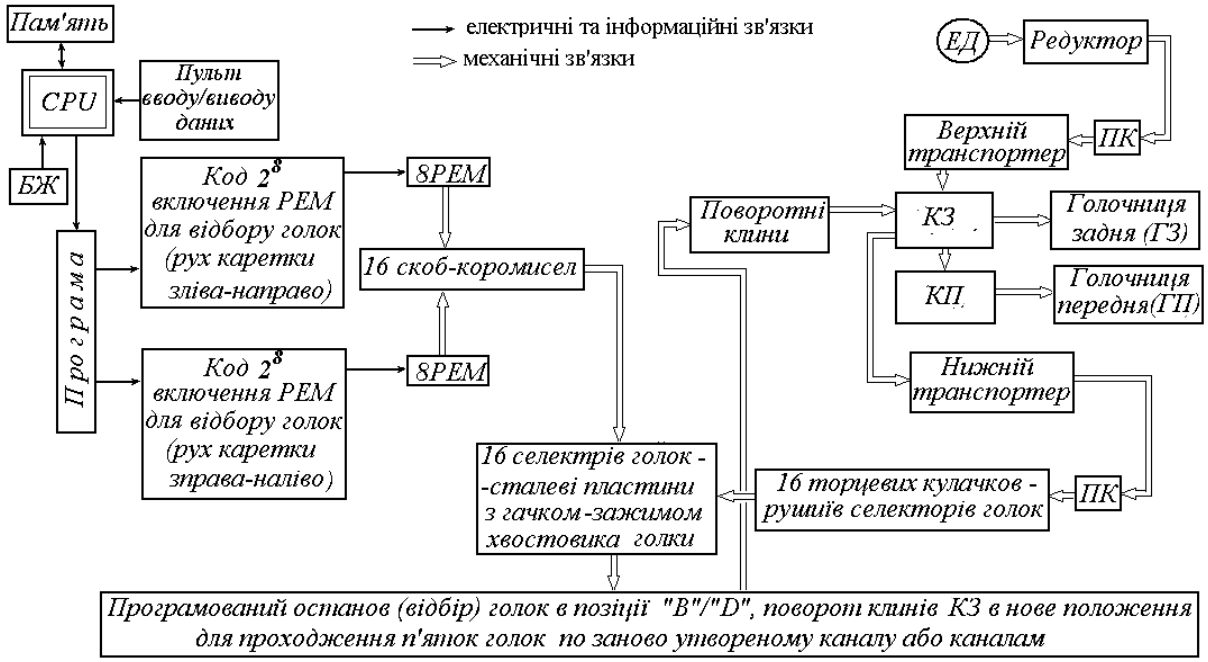


Рис. 2.4. Функціональна схема комп'ютерної плоско в'язальної машини моделі КН 965і Brother (Японія) з електромагнітним відбором голок

Комбінована схема комп'ютерної плосков'язальної машини наведена на рис. 2.5, де прийняті наступні умовні позначення елементів схеми:

ЕД - електродвигун і редуктор ***Р*** – для створення зворотно-поступального руху в'язальної каретки;

S/S – вмикач електропривода в'язальної каретки;

ВТ і ***НТ*** – верхній транспортер і нижній транспортер;

КЗ і ***КП***, ***ГЗ*** і ***ГП*** – карета задня і каретка передня, голочниця задня і голочниця передня;

РЕМ 1…РЕМ 16 – шістнадцять (два байта) ***Р****еле* ***Е****лектро****М****агнітних* (міні-соленоїдів, надалі прийнято умовне скорочення *«електромагніти»*) ;

К1…К16 – шістнадцять торцевих кулачків по одному на кожну пластину-селектор голок;

CPU – (***C****entral* ***P****rocessing* ***U****nit*) мікроконтролер;

БЖ – блок електричного живлення;

1,2,7,8– шківи з кілками;

3 - ведуча пружна защіпка головної каретки (закріплена на верхньому транспортері *ВТ*;

4 – ведена защіпка, яка закріплена на каретці;

5 – тримач-затискач для з'єднання каретки задньої (додаткової) *КЗ* і каретки головної *КП*;

9 – конічні передачі верхнього і нижнього транспортерів;

10 – вал з кулачками *К1…К16;*

11 – коромисло для зворотно-поступального руху пластини-селектора голок з автоматичним гальмуванням від реле електромагнітного;

12 – зворотна пружина.

Після заправлення машини пряжею і введення даних програми в'язання рисунка натисканням клавіші S/S включається електродвигун *ЕД*, який за допомогою редуктора *Р* і системи керування приводом (на рис. 6.10 не зображена) перетворює обертальний рух ротора *ЕД* в зворотно-поступальний рух верхнього транспортера *ВТ,* а далі цей рух передається по кінематичному ланцюгу згідно з наступним виразом:

$$\begin{array}{l} ВТ \Rightarrow 3 \Rightarrow 4 \Rightarrow КЗ \Rightarrow 5 \Rightarrow КП \\ \qquad\qquad \llcorner\!\!\!\rightarrow НТ \Rightarrow 9 \Rightarrow 10 \Rightarrow К1 \ldots К16 \rightarrow пластини-селектори\ голок \end{array} \qquad (2.0)$$

Стрічки транспортерів *ВТ* і *НТ* виконані зі сталевої стрічки перфорованої отворами з постійним кроком по всій довжині. В ці отвори входять кілки шківів *1,2,7,8.* Сталева стрічка не розтягується при експлуатації машини і не проковзує на шківах. Сталий зворотно-поступальний рух стрічок обох транспортерів і в'язальної каретки відносно нерухомих голочниць *ГЗ* і *ГП* за допомогою конічної зубчатої передачі 9 перетворюється спочатку в зворотно-обертальний рух валу 10 і торцевих кулачків *К1…К16,* а потім знову в зворотно-поступальний рух, але вже останнього елемента у виразі (2.0), а саме пластин – селекторів голок з крючками-захватами голок (рис. 2.5) для програмованого зупинення голок у відповідному голковому пазу головної (передньої) голочниці (фонтури) *ГП.*

Кількість електромагнітів в машини дорівнює шістнадцяті, з яких 8 електромагнітів служать для відбору голок при русі каретки зліва направо і 8 служать для відбору голок при русі каретки справа наліво. Тобто кількість електромагнітів обрана такою, що дорівнює двом байтам (8х2=16 розрядів), по одному електромагніту у відповідному розряді одного байту. Електромагніти спрацьовують від 8-розрядного коду при русі каретки вправо і вліво. Кожний електромагніт (*РЕМ*) може бути знеструмленим і виключеним (стан «0») або включеним (стан «1»). Тому при восьми електромагнітах можна перетворити 8-розрядний програмний код, що складається з «0» і «1» для включення/виключення голок в кожному петельному ряді при в'язання запрограмованого рисунку та запрограмованого типу переплетення. Всього 8-розрядних кодів у двійковій системи відліку може бути $2^8 = 256$, що більше числа 200 – кількості трикотажних голок в кожній фонтурі (голочниці) машини.

При 7 електромагнітах кількості двійкових кодів була б недостатня для включення/виключення 200 голок машини, оскільки $2^7 = 127 < 200$ голок в голочниці.

Згідно з виразом (2.0) відповідні програмному коду крючки-захвати пластин – селекторів голок (рис. 2.5) зупиняють відповідні трикотажні голки за рахунок їх притискання до відповідного голкового паза головної (передньої) голочниці *ГП.*

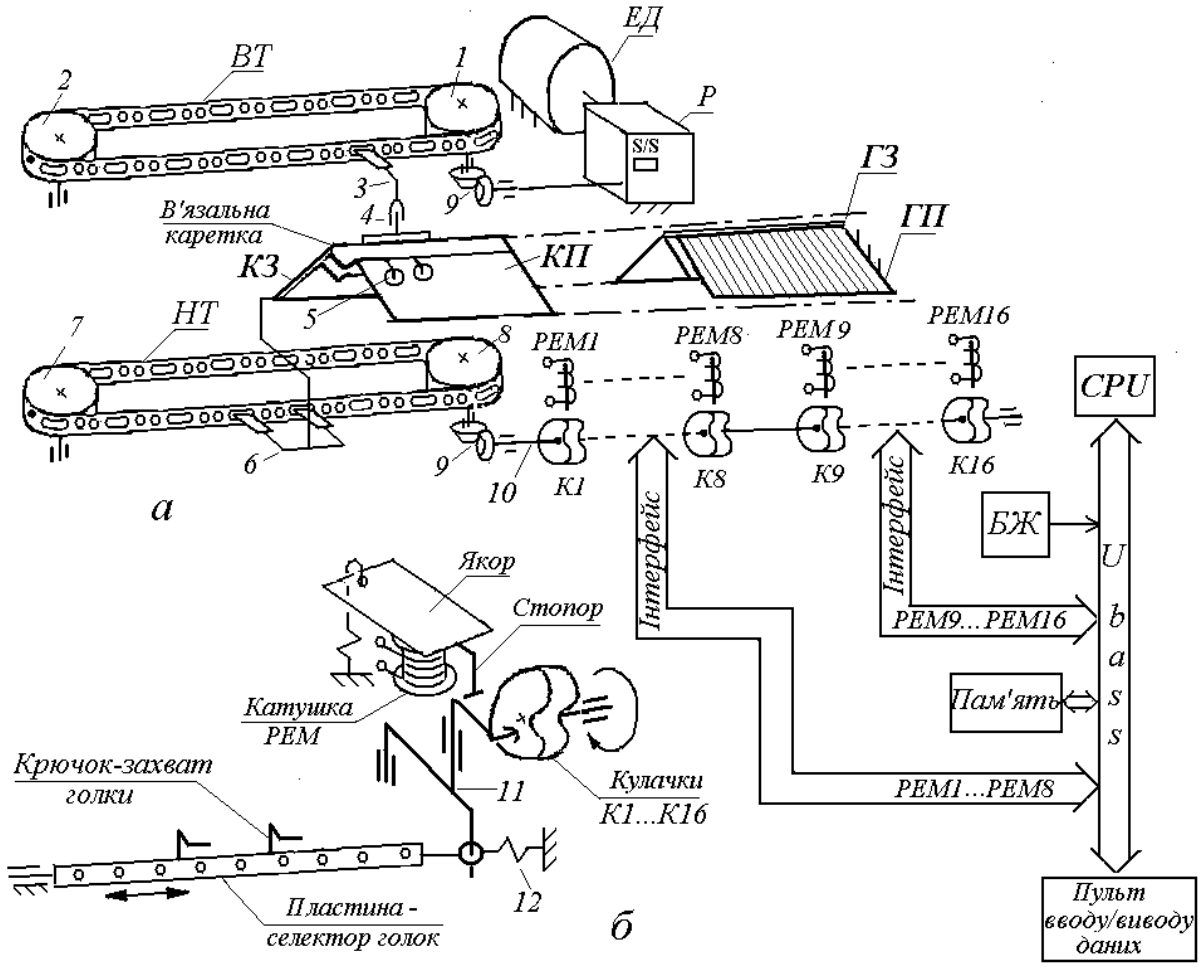


Рис. 2.5. Комбінована схема комп'ютерної плоско в'язальної машини моделі KH-965i Brother (Японія): *а* – з'єднання в'язальної каретки з механізмом селекції голок; *б* – розгорнута схема одного з шістнадцяти механізмів селекції голок

2.2. **Жорстка системи керування циклом (1→N1)·n з програмоносієм типу «багатокроковий кулачок» машин легкої промисловості**

Жорстка система керування машинами – це система керування типу ***«Розподільний Вал»*** (типу ***РВ***) технологічними машинами, у який закон руху робочих органів механізмів задається програмо носіями, які є ведучими ланками у вигляді кривошипу або ексцентрику, які закріплені на головному валі і які задають програму роботи (функцію положення) за цикл що дорівнюється 1 обороту головного валу. Багато циклові жорсткі програмоносії у вигляді багатокрокових кулачків (копірних дисків) змонтовані на

додатковому валі, який кінематичне з'єднаний з головним валом передаточним відношенням. Для зміни програми роботи кривошипа і ексцентрика потрібна відкрутка для послаблення кріплення і наступного його повороту на певний кут відносно головного валу в нове положенні. Для зміни програми потрібно або переключити механізм на роботу с іншим кулачком, або замінити програмоносій на програмоносій з іншою програмою.

В літературі, яка присвячена проектуванню швейних машин-напіавтоматів кулачкові програмоносії («копіри») циклових швейних машин-напівавтоматів по суті справи не розглядалися. Намагання деяких авторів подати методику їх проектування з використанням відомих «загально-інженерних» методів є вкрай помилковими. У загально-технічній літературі з проектування кулачкових механізмів ці унікальні кулачки, їх структура, особливості способу їх проектування теж не знайшли відображення. В роботі [10] автори запровадили терміни: «багатокрокові» та «крокові-ступінчасті» кулачки, щоб відокремити ці особливі кулачки-програмоносії від звичайних - одно крокових.

Багатокрокові та крокові-ступінчасті кулачкові програмоносії швейних машин-напівавтоматів містять декілька десятків фазових кутів-кроків кожен з яких, як окремий кулачок, забезпечує коромисловому штовхачу рух типу «вистій-переміщення-вистій» або «вистій-переміщення» - в залежності від прийнятого відліку фаз-кроків (рис. 2.6).

У більшості випадків фазові переміщення $\delta = R_2 - R_1$ розташовуються відносно центра О обертання програмоіносія ступенчато – мають різні значення радіусів їх основного кола R_1 та різну величину фазових переміщень δ, що обумовлено рапортом строчки її конфігурацією, співвідношенням довжини стібків, що утворюють строчку.

В основу розрахунків та побудови профілю багатокрокових та крокових-ступінчатих кулачкових програмоносіїв покладена зовсім інша ніж зазвичай концепція, відповідно, використовується особливий, унікальний спосіб та алгоритм їх структурного та метричного синтезу. Особливості проектування кулачкових програмоносіїв починаються з необхідності визначення (вибору) взаємопов’язаних вихідних даних які складають тріаду параметрів $d = 2r = \rho$, що визначає, співвідношення: ***діаметра ролика*** d коромислового штовхача; ***радіусів*** r – дуг окружностей центрового профілю; ***радіусів*** ρ – дуг окружностей еквідистант, що окреслюють робочий профіль кожної окремої фази руху програмоіносія. Усвідомлений вибір значення тріади параметрів передбачає знання впливу її величини на габарити копірного диска, як

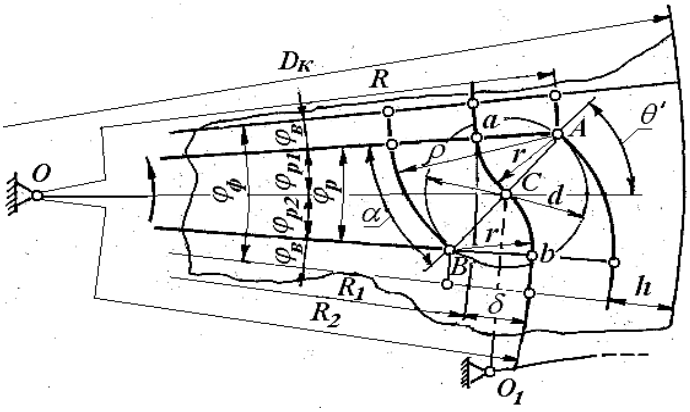
специфічного вибору механізму та на функціональні й динамічні характеристики програмо- носія, як ведучої ланки механізму коливання голки або переміщення матеріалу.

З розрахункової схеми (рис. 2.6,а) випливають залежності:

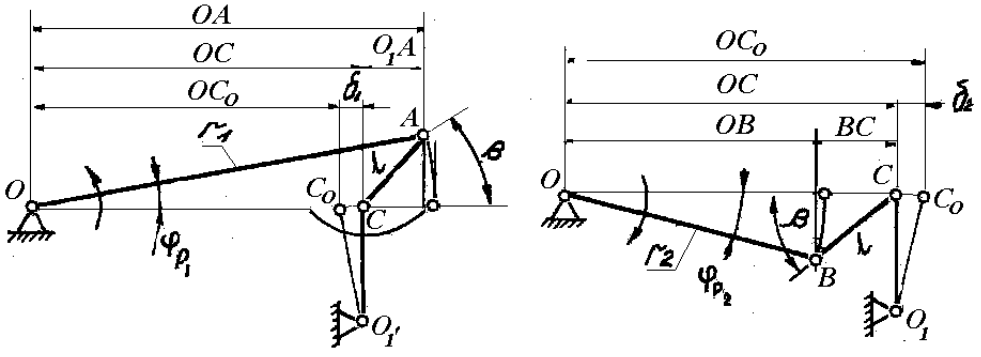
$$\delta = R_2 - R_1 \approx d(1 - cos\theta'); \quad d \approx \frac{\delta}{1-cos\theta'} \tag{2.1}$$

та $$OA = R = R_1 + r \approx \frac{d \cdot sin\theta'}{sin\varphi_p} \tag{2.2}$$

де θ' - прийнята величина номінального кута тиску θ, що характеризує крутизну центрового профілю та визначає прирощення $\delta = R_2 - R_1$ радіуса R_1 основного кола. Оскільки кут тиску θ звичайно не беруть більшим за 45^o то $\delta_{max} \approx d(1 - cos\theta') < 0{,}3d$. При зменшенні значень d та кута θ' суттєво зменшується головна функціональна характеристика кулачка δ. Наприклад, при $\theta = 30'$ $\delta \approx 0{,}12\,d$, а при $\theta = 20'$ $\delta \approx 0{,}06\,d$.



а



б **в**

Рис. 2.6. Схема (а) типового фазового кута – кроку кулачкового програмоносія та еквівалентні заміняючи механізми: б - механізм еквівалентний увігнутої частині «aC» профілю «aCb» програмоносія; в - механізм еквівалентний опуклої частини профілю «Cb» профілю «aCb» програмоносія

Залежність δ, та R від d, θ та значення $\varphi_p(\varphi_p \approx 4', \ \varphi_p \approx 6')$ при $\varphi_p \approx 8'$ наведені на рис. 2.7.

Що до параметрів R, який обумовлює габарит копіру $(D_k \approx 2R + r)$ його величина зростає зі збільшенням значень d і θ та зменшенням величини фази руху φ_p, особливості визначення якої розглянуто нижче.

Залежності (2.1), (2.2) які відображені в діаграмі (рис. 2.7) свідчать, що значення $\delta = 2{,}0 \ldots 3{,}5$ мм., необхідні для швейних машин-напівавтоматів, при порівняно невеликих габаритах копіру, $(R \approx 75 \ldots 80$ мм, $D_k \approx 175)$ можна одержати при величині $d \approx 11 \ldots 13$ мм., та куті тиску θ' близькому до гранично допустимого $\theta q = 45'$. Лише при таких значеннях d і θ, та, додатково, використанні передаткового відношення $\text{і} = \frac{\delta}{t} \approx 0{,}7 \ldots 0{,}4$, відповідних механізмів (переміщення матеріалу, або попереднього коливання голки, можна одержати стібки довжиною $t = 4 \ldots 8$ мм, необхідні для виготовлення закріпок, пришивання фурнітури і таке інше.

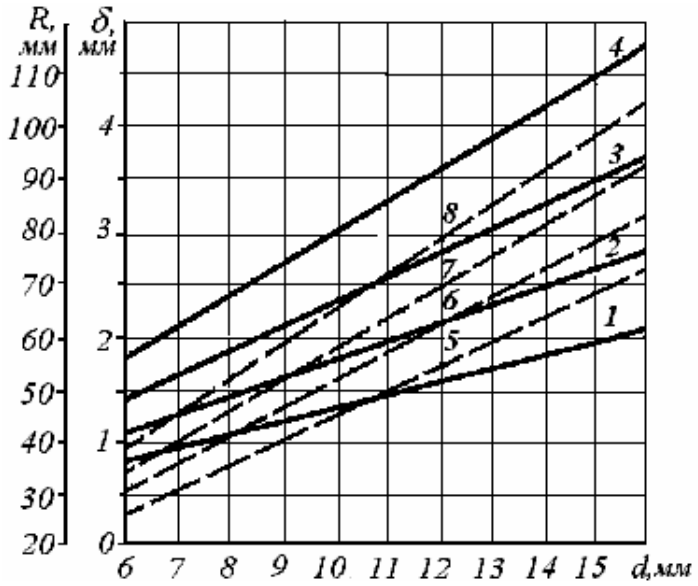
Вочевидь прийняте у практиці швейного машинобудування значення тріади $d = 2r = \rho = 12$ мм є цілком обгрунтованим і раціональним. Зважаючи на те, що технологія виготовлення кулачкових програмоносіїв, яка в усіх її аспектах удосконалювалася на протязі багатьох десятиріч, зорієнтована саме на цю величину, змінювати її, без важливих на те причин недоцільно.

Вибір величини тріади параметрів в її органічному зв'язку зі значенням кута тиску θ і величиною δ забезпечує можливість розрахунку усіх параметрів першої визначальної фази-кроку кулачкового програмоіносія.

Наприклад, при прийнятих значеннях: $d = 2r = \rho = 12$ мм; $\delta = 3$ мм; $\varphi_p = 6°17'$; з теореми косинусів визначають довжину радіусу $R_1 = R_{min}$ основного кола центрового профілю.

$$D^2 = (R_1 + 6)^2 + (R_1 - 3)^2 - (R_1 + 6)(R_1 - 3) \cdot 2cos\,\varphi_\text{p} \qquad (2.3)$$

Після перетворень вираз (2.3) набуває вигляду квадратного рівняння



$$aR^2 + bR_1 - C = 0,$$

де $a \approx 0{,}01202;$

$b \approx 0{,}03606;$ $c \approx 63{,}216$

Рис. 2.7. Залежність δ та R від значення d при різної величині кута тиску θ : *1*, *2*, *3*, *4* – $\delta(d)$, відповідно, при $\theta = 30,\ 35,\ 40,\ 45^0$; *5*, *6*, *7*, *8* – залежність $R(d)$ відповідно, при $\theta = 30,\ 35,\ 40,\ 45^0$

Використовуючи, формулу рішення загального виду

$R_1 \approx \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ одержуємо:

$$R_1 = \frac{-0{,}03606 + \sqrt{0{,}0013 + 4 \cdot 0{,}0120 \cdot 63{,}216}}{0{,}02404} \approx 71{,}036 \approx 71 \text{ мм}$$

Потім визначають інші параметри профілю: $R_2 = R_1 + \delta = 74;\ R = R_1 + r = 77$ мм; $AB = d = 12$ мм, $\rho = d = 12$ мм.

Кожну фазу-крок багатокрокового або крокового-ступінчастого програмоносія можна розглядати, як окремий механізм із вищою IV класу) кінематичною парою, яку утворює ролик d коромислового штовхача O_1C з пазом. Оскільки (рис. 2.6) центровий профіль паза окреслено двома однаковими дугами радіуса r окружності d з центрів A та B то такий кулачковий механізм, при дослідженні, можна замінити двома еквівалентними важільними механізмами. Кожний з двох частин центрового профілю (рис.2.6) увігнутої aC, та опуклої Cb, що окреслені дугами окружності $r \approx 6$ мм. З точок A та B, відповідає свій еквівалентний замінюючий механізм. Для частини профілю aC еквівалентним заміюючим механізмом є центральний коромисловий чотириланковик $OACO_1$ (рис.2.6,а) з параметрами: $OA = r_1 = 77$ мм; $AC = \ell = 6$ мм; $CO_1 = 50 \ldots 70$ мм. Для частини Cb центрового профілю, еквівалентним замінюючим механізмом є центральний коромисловий чотириланковик

$OBCO_1$ (рис.2.6,б) з параметрами: $OB = r_2 = 68$ мм, $BC = \ell = 6$ мм; $CO = 50 \ldots 70$ мм.

Оскільки за алгоритмом розрахунків передбачається вибір величини переміщення δ мм, що є хордою дуги, яку окреслює центр ролика та функцію положення $\delta_1 = f(\varphi)$ доцільно виражати саме хордою, маючи на увазі, що при звичайних значеннях довжини коромислового штовхача $L \approx 50 \ldots 70$ мм та хордах $\delta = 2 \ldots 3$ мм довжина відповідних дуг, при розрахунках навіть при використанні $\pi \approx 3{,}14159$, виявляється меншою, або однаковою з довжиною їх хорд.

Тому одержані, замінюючи механізм доцільно вважати центральними коромислово-повзунними механізмами.

В структурному сенсі механізм 2.6, а є аналогом центрального кривошипно-повзуного механізму з шатуном спрямованим вниз. Він характеризується

$$\lambda = \frac{r_1}{\ell} = \frac{77}{6} \approx 12.833$$

і є еквівалентним в межах обертання кривошипу на кут $0^o \leq \varphi \leq \varphi_{p_1}$.

Замінюючий механізм 2.6,б є аналогом «оберненого» кривошипно-повзуного механізму [15] з шатуном спрямованим вверх. Він характеризується

$$\lambda = \frac{r_2}{\ell} = \frac{68}{6} \approx 11.333$$

і є еквівалент в межах обертання кривошипу на кут $0^o \leq \varphi \leq \varphi_{p_2}$.

З умови виключення жорстких ударів на початку та в кінці руху коромислового штовхача центр A дуги r окружності розташоване (рис. 2.6) на прямій, що проходить через точки O та a, відповідно, центр B другої дуги r окружності розташовано на прямій, що проходить через центр O та точку b. Умові, що забезпечує відсутність жорсткого удару при переході від прискореного руху на увігнутої частини aC центрового профілю до уповільненого руху на опуклої Cb його частині відповідає розташування точки Cспряження дуг всередині відрізка $AB = d$.

При цьому кути обертання кулачка φ_{p_1} та φ_{p2}, необхідні, зокрема, для побудови й дослідження еквівалентних замінюючих механізмів, виявляються різними. Радіус-вектор OC, що визначає положення точки C

перегину центрового профілю та поділяє фазу руху φ_p на два нерівних кути - φ_{p_1}, φ_{p_2} визначають з теореми косинусів:

$$OC = \sqrt{OA^2 + Ac^2 - 2OA \cdot AC \cdot cos\alpha} \approx 72{,}39$$

де $cos\alpha = \frac{OA^2 - AB^2 - OB}{2 \cdot OA \cdot AB} = 0{,}7841$ і $\alpha = arccos\, 0{,}7841 = 38°22'$

кут $\varphi_{p1} = arccos\, \frac{OA^2 + OC^2 - AC^2}{2OA \cdot OC} \approx arccos 0{,}99867 \approx 2°57'$

кут $\varphi_{p2} = \varphi_p - \varphi_{p1} \approx 6°17' - 2°57 \approx 3°20'$ або

$$\varphi_{p2} = arccos\, \frac{OC^2 + OB^2 - BC^2}{2OC \cdot OB} \approx arccos 0{,}9983 \approx 3°20',$$

що підтверджує коректність розрахунків.

Таким чином, замінюючий механізм $2.6, б$ є еквівалентним кулачковому в межах повороту коромисла $r_1 = 77$ мм, на кут $0 \le \varphi \le 2°57'$ що відповідає повороту кулачка на кут $0 < \varphi_{p1} \le 2°57'$. Замінюючий механізм $2.6, в$ є еквівалентним кулачковому в межах повороту коромисла $r_2 = 68$ мм на кут $0 \le \varphi \le 3°20'$, що відповідає повороту кулчка від $\varphi_{p_1} = 2°57'$ до $\varphi = \varphi_{p_1} + \varphi_{p_2} = 6°17'$.

Функцію положення $\delta_1 = f(\varphi_{p_1})$ механізму $3a$ визначають з геометричних співвідношень його ланок:

$$\delta_1 = C_o C = OC - OC_o,$$

де $OC = OA - AC = r_1 cos\varphi_{p_1} - \ell cos\beta$;

$OC_o = r_1 - \ell$, звідки

$$\delta_1 = \ell(1 - cos\beta_1) - r_1(1 - cos\,\varphi_{p_1} - \ell cos\varphi_{p_1});$$

використовуючи $\lambda_1 = \frac{r_1}{\ell}$ ($\ell = \frac{r_1}{\lambda}$), отримаємо:

$$\delta_1 = r_1 \left[\frac{1}{\lambda_1}(1 - cos\beta_1) - \left(1 - cos\varphi_{p1}\right)\right] \qquad (2.4)$$

Враховуючи залежність (2.4) та тригонометричну тотожність $cos\beta_1 = \sqrt{1 - sin^2\beta_1}$ одержують

$$\delta_1 = \left[\frac{1}{\lambda_1}\left(1 - \sqrt{1 - \left(\lambda_1 sin\varphi_{p1}\right)^2} - \left(1 - cos\varphi_{p1}\right)\right)\right]$$

Диференціюючи функцію положення (2.4) одержують вирази для визначення швидкості та прискорення:

$$V_1 = -r_1\omega(cos\varphi_{p1}tg\beta_1 - sin\varphi_{p1}) \tag{2.5}$$

$$\alpha_1 = -r_1\omega^2\left(sin\varphi_{p1}tg\beta_1 + \lambda\frac{cos^2\varphi_{p1}}{cos^3\beta_1} - cos\varphi_{p1}\right) \tag{2.6}$$

У виразах (2.4)…(2.6), $B = arcsin(\lambda_1 sin\varphi_{p1})$,

де $\lambda_1 = \frac{r}{\ell} = \frac{77}{6} \approx 12{,}83/$

З рис. 2.6, б визначають функцію положення замінюючого механізму при обертанні коромисла $r_2 = 68$ мм в межах кута $0^o \leq \varphi_{p2} \leq 3^o20'$.

З геометричних співвідношень $\delta_2 = C_0C = OC_0 - OC$,

де $OC_0 = r_2 + \ell$

$$OC = OB + BC = rcos\varphi_{p2} + \ell cos\beta_1$$

$$\delta_2 = \ell(1 - cos\beta_1) + r_1(1 - cos\varphi_{p2}).$$

Використовуючи співвідношення $\lambda = \frac{r_2}{\ell}$ ($\ell = \frac{r_2}{\lambda_2}$)

$$\delta_2 = r_2\left[\frac{1}{\lambda^2}(1 - cos\beta_2) + \left(1 - cos\varphi_{p2}\right)\right] \tag{2.7}$$

Ураховуючи залежність (2.7) та використовуючи тригонометричну тотожність $cos\beta_2 = \sqrt{1 - sin^2\beta_2}$ і диференціюючи функцію положення (2.7) одержують наступні вирази для визначення швидкості та прискорення:

$$V_2 = -r_2\omega(cos\varphi_{p2}tg\beta_2 - sin\varphi_{p2}) \tag{2.8}$$

$$a_2 = -r_2\omega^2\left(sin\varphi_{p2}tg\beta_2 + \lambda\frac{cos^2\varphi_{p2}}{cos^3\beta_2} - cos\varphi_{p2}\right) \tag{2.9}$$

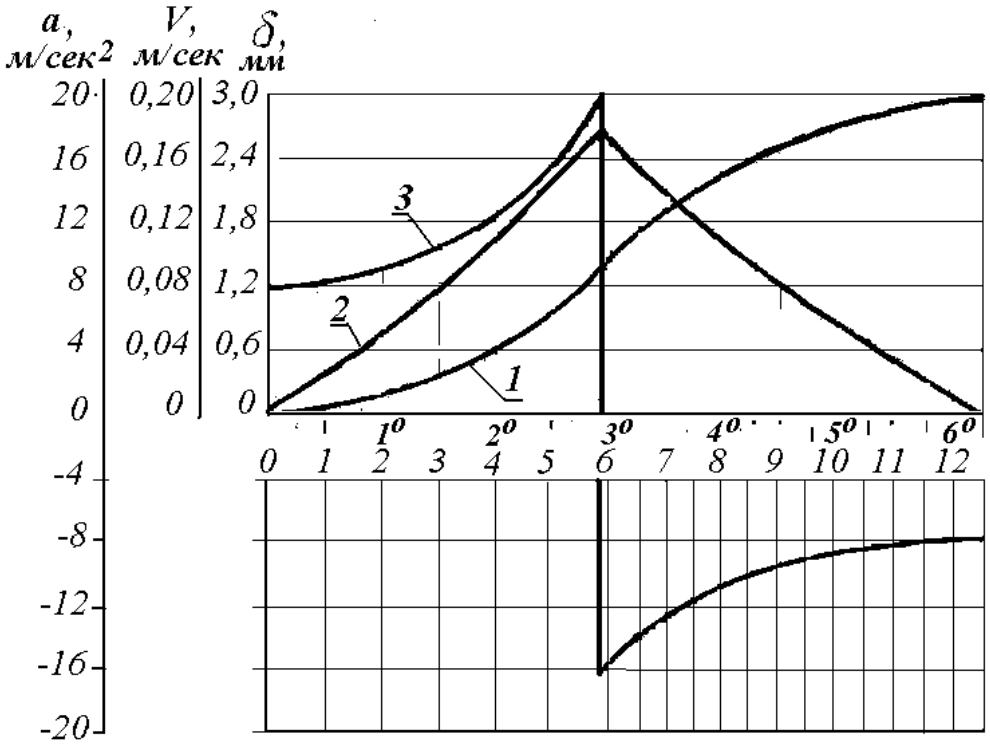


Рис. 2.8. Кінематичні характеристики руху штовхача кулачкового програмоносія: 1 – переміщення; 2 – швидкість; 3 – прискорення

У виразах (2.7)…(2.9) кут $\beta = \arcsin(\lambda_2 sin\varphi_{p2})$ де $\lambda = \frac{68}{6} \approx 11{,}33$.

По виразам (2.4)…(2.6) одержані значення переміщення, швидкості, прискорення веденої ланки механізму (рис.2.6, б) при повороті ведучої ланки на кут, $\varphi_{p1} = 0^o30', 1^o00', 1^o30', 2^o00', 2^o30', 2^o57'$ *для побудови характеристики увігнутої частини aC профілю aCb програмоносія.* По виразам (2.7)…(2.9) одержані значення переміщення (з крайнього верхнього положення) веденої ланки механізму (рис.2.6, в) при повороті ведучої ланки на кут $\varphi_{p2} = 0^o30', 1^o60', 1^o30', 2^o00', 2^o30', 3^o00', 3^o20'$ *для побудови характеристики опуклої частини профілю Cb профілю aCb програмоносія.*

При розрахунках прийнято: частота обертання головного валу $n = 200$ хв$^{-1}$; передаткове відношенням між головним валом і програмоносієм $i = 42$; $d = 2r = \rho = 12$ мм; $\varphi_p = 6'17$; $\varphi_{p_1} = 2'57$; $\varphi_{p_2} = 3'20$. По одержаним даним побудована кінематична діаграма типової фази-кроку кулачкового програмоносія швейної машини (рис. 2.8).

Проведені дослідження дозволяють вирішити першорядне важливе питання проектування кулачкових програмоносіїв, що до визначення доцільного співвідношення фази руху φ_p та фази вистою $\varphi_в$, при відомому, значенні відомого кута φ_p . Відомо, що у швейних машинах кут повороту головного валу у період руху вістря голки «над матеріалом» та кут його повороту при русі голки «в матеріалі» приблизно однакові. З цього випливає формально необхідне співвідношення фази руху та фази вистою - $\varphi_р \approx \varphi_в$. Функції положення (2.4) і (2.7) та діаграма (рис.2.8) свідчать, що на початку і в кінці фази руху, при повороті ведучих ланок r_1 та r_2 на кут $\Delta\varphi \approx 1^o$ відбувається порівняно мале переміщення ведучої ланки - $\Delta\delta \approx 0{,}10 \ldots 0{,}14$ мм.

Ураховуючи передаткові відношення відповідних механізмів $i \approx 0{,}7 \ldots 0{,}4$ це відповідає переміщенню матеріалу (або голки) на величину $\Delta t \approx \Delta\delta/i = 0{,}25 \ldots 0{,}5$ мм. Такі переміщення у період вистою вважаються допустимими, тому ці дві частини фази руху $(2\,\Delta\varphi)$ умовно відносять до фази вистою, збільшуючи, відповідно, розрахункову величину фази руху до величини:

$$\varphi_р \approx 0{,}5\,\varphi_ф + 2\,\Delta\varphi \approx 4^o17' + 2^o \approx 6^o17'$$

Цей прийом дозволяє суттєво поліпшити конструктивні та динамічні характеристики копіру та механізмів, в яких він служить ведучою ланкою. У разі не використання цього прийому - застосуванні формально-необхідної величини $\varphi_р$, наприклад, $\varphi_p \approx 4^o17'$ параметри R_1, R_{max}, $D_к$ суттєво збільшуються і досягнуть, відповідно, значень: $R_1 \approx 104$, $R \approx 110$, $D_к \approx 240$. Швидкість V_{max}, згідно з виразом (2.5) збільшиться до $0{,}26$ м/сек., прискорення згідно з (2.6) стає вдвічі більшим $(a \approx 40$ м/сек$^2)$.

Вочевидь прийом, що передбачає збільшення розрахункової величини фази руху до значень, що становить 65…70% від величини фазового кута – або в $2{,}5 \ldots 2{,}7$ рази більшими за фазу вистою, є ще однією з особливостей способу проектування багатокрокових кулачків-програмоносіїв швейних машин.

Таким чином, на підставі аналізу одержаної кінематичної діаграми можна скласти об’єктивне уявлення про кінематичні й динамічні характеристики багатокрокових кулачкових програмоносіїв. Діаграма переміщення, що складається з $\delta_1 = f(\varphi_{р1}), \delta_2 = f(\varphi_р)$, на початку і в кінці плавно сполучена з дугами окружностей R_1, R_2, що забезпечує

нульові значення швидкості, але в точці перегину графіка переміщення, що збігається з точкою (спряження дуг центрового профілю, на діаграмі) $V(\varphi)$ утворюються характерне загострення, що відображує миттєву зміну величини швидкості. Відповідно, діаграма прискорення $a(\varphi)$ характеризується розривами (стрибками) у точках спряження різних по характеру або величині дуг окружностей, що утворюють центровий профіль. Таким чином, динамічна характеристика розглянутого кулачкового програмоносія не є бездоганною, але зважаючи на реальні дуже малі значення швидкості та стрибків прискорення ($V_{max} \approx 0{,}19$ м/сек, $a_{max} \approx 20$ м/сек^2), що є вкрай незначними, що свідчить про високий рівень надійності і робото здатності крокових кулачкових-програмоносіїв при використанні їх в швейних машинах.

Більшість виконавчих механізмів швейних машин працюють в умовах значно більших швидкостей та прискорень. Так механізм голки тієї ж машини 220кл ОЗШМ виконує свої функції при $V_{max} \approx 4$ м/с і $a_{max} \approx 645$ м/сек^2 , що в двадцять та в тридцять разів більші ніж значення V та a, що характеризують механізм з кроковим програмоіносієм.

До технологічних машин із жорсткою системою керування з програмоносієм типу «багатокроковий кулачок» відносяться, наприклад всі циклові швейні напівавтомати для пришивання фурнітури і виготовленні петель і закріпок на одязі, плосков'язальні напівавтомати типу ПВК з функціональної групою механізму додавання голок і механізм автоматичної збавки петель, а також кулачковий механізм зсуву задньої голочниці та інші.

2.3. **Гнучкі системи керування в обладнанні галузі і машинобудуванні**

2.3.1. **Технологічні машини легкої промисловості з мехатронним керуванням**

На рис. 2.9 у вигляді принципових схем наведені приклади застосування в швейних (мехатронних) машинах мікропроцесорного (електронного) керуванням.

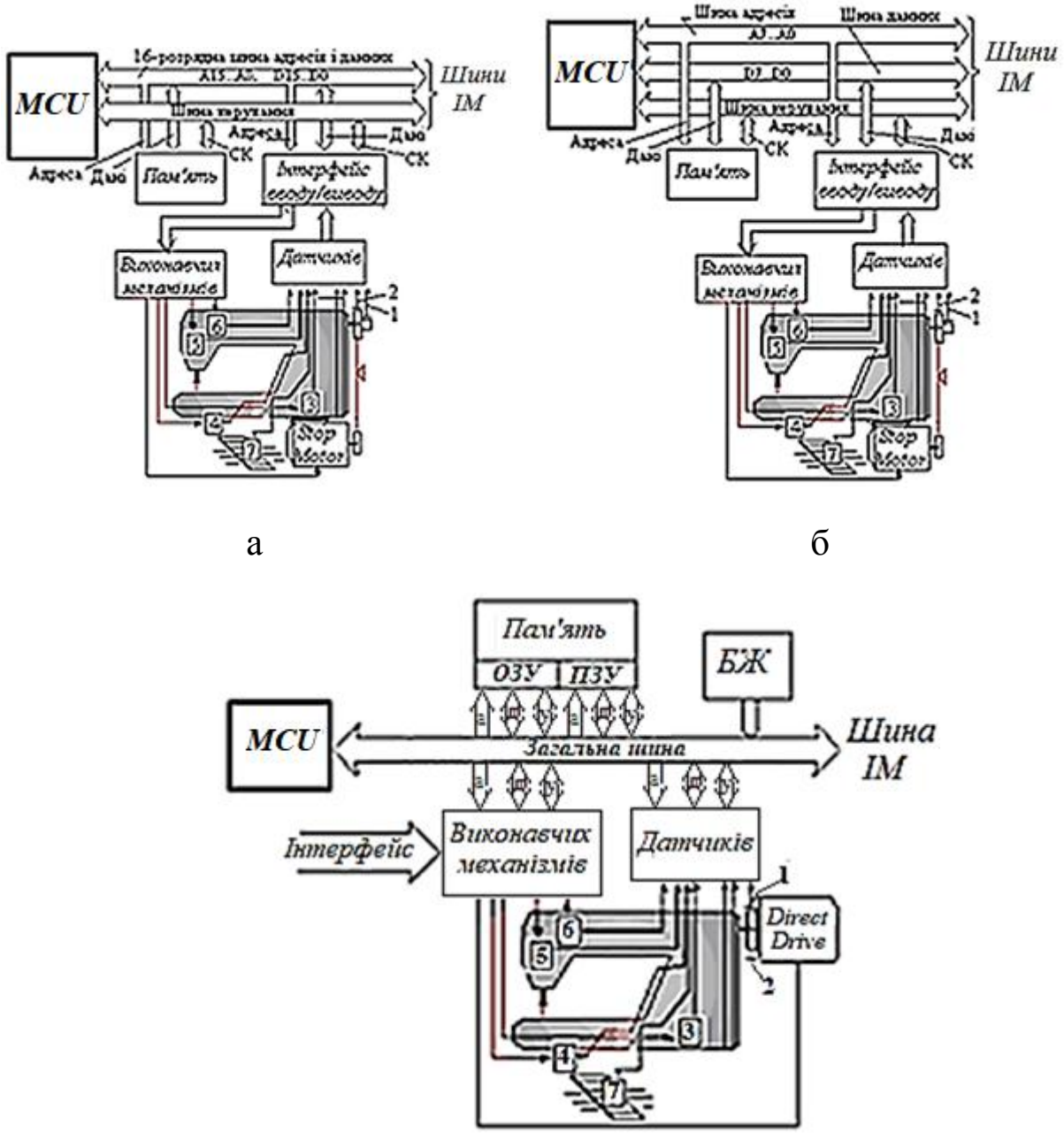


Рис. 2.9. Приклади застосування мехатроніки в швейних машинах: а – із загальною шиною адресів (А) і даних (Д) та окремою шиною сигналів керування (У); б – з окремими шинами А, Д і У сигналів керування ; в – із загальною шиною А, Д і СК сигналів керування

В таблиці 2.1 наведені приклади програмно керованих механізмів (ПКМ) обладнання легкої промисловості з ЦПК і ЧПК.

Таблиця 2.1 ПКМ автоматизованих швейних машин

Швейні машини з гнучким керуванням	*Програмно керовані механізми* (ПКМ)							Кількість швейних головок в одному автоматі	Кількість голок в одної швейної голов-ці
	ПКМ	ПКМ крокових 2D-переміщень об'єкту обробки	ПКМ обрізки ниток	ПКМ регулятора натягу голкової нитки	ПКМ крокового переміщення каретки з голко-водами	ПКМ вибору голково-ду	ПКМ затискання кінців обрізаних ниток		
ЦКП	+	+	+	+	-	-	-	1	1
ЧПК	+	+	+	+	+	+	+	1...20	1...15

На рис.2.10 наведена принципова схема швейно-вишивальної CNC-машини, де прийняти наступні умовні позначення:

МСU – мікроконтролер;

ДЖ – джерело живлення;

УВВ – устрій вводу/виводу;

ЕК – електронний комутатор;

ДШ – дешифратор команд;

БПП – буферний підсилювач потужності;

БЕФ – блок електронного форсування;

КДх, КДу – крокові електродвигуни.

Мікронтролер - це електронний пристрій за архітектурою комп'ютера. На відміну від мікропроцесора мікроконтролер орієнтований на виконання конкретних завдань, а саме – гнучкого програмного керування засобами автоматизації технологічних машин і апаратів, верстатів та інших технічних систем.

Мікропроцесорні (електронні) системи керування це контролери, які вбудовані у технологічні машини для механізмів, що кінематичне не з'єднані з головним валом і які за структурою є міні комп'ютером, який орієнтований на виконання конкретних технічних завдань. Комп'ютери орієнтовані на виконання загальних задач (моделювання, розрахунків, оптимізації, статистики та інших). І поширена назва таких автоматизованих технологічних машини – комп'ютерні швейні машини, комп'ютерні в'язальні машини тощо випливає з принципу узагальненості структур побудови контролера і комп'ютера. Але в таких автоматизованих

технологічних машинах залишається для деяких механізмів або їх функціональних груп жорстка система керування типу «*розподільний вал*».

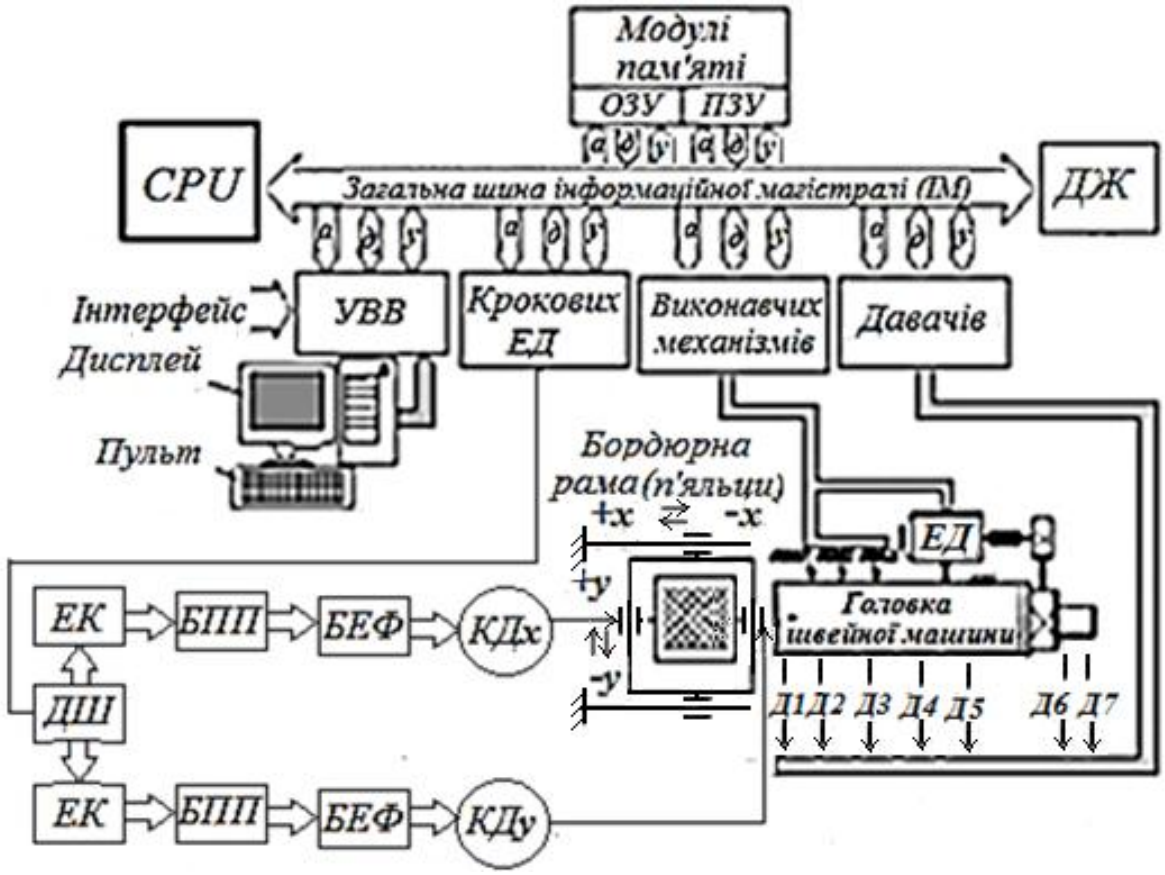


Рис. 2.10. Принципова схема швейно-вишивальної CNC-машини

2.3.2. **Приклад циклової системи керування верстатами машинобудування**

На рис. 2.11 наведені наступні умовні позначення елементів мехатроніки:

1A1, 2A1 і 3A1 – виконавчі пневмоциліндри;

1B1 і 1B2, 2B1 і 2B2, 3B1 і 3B2 – датчики крайніх положень;

B4 і B5 – датчики правильного положення деталей на позиції свердлення і на позиції зенкування/

Початкові умови аналізу циклового програмного керування: Деталі доставляються стрічкою конвеєра на станцію з двома постами для механічної обробки деталей. Як тільки деталі досягають робочого місця,

включаються операції свердлення і зенкування. Для керування цими механізмами використовуються два пнемо циліндри 1А1 і 2А1. Конвеєр позначається однією робочою позицією як третій циліндр 3А1. Для визначення правильного положення деталей при свердленні і зенкуванні встановлено два датчики В4 і В5. Глибина свердлення фіксується датчиками кінцевого положення 1В2 і 2В2. Початкові положення циліндра конвеєра, свердла і розгортки встановлюються датчиками 1В1, 2В1 і 3В1.

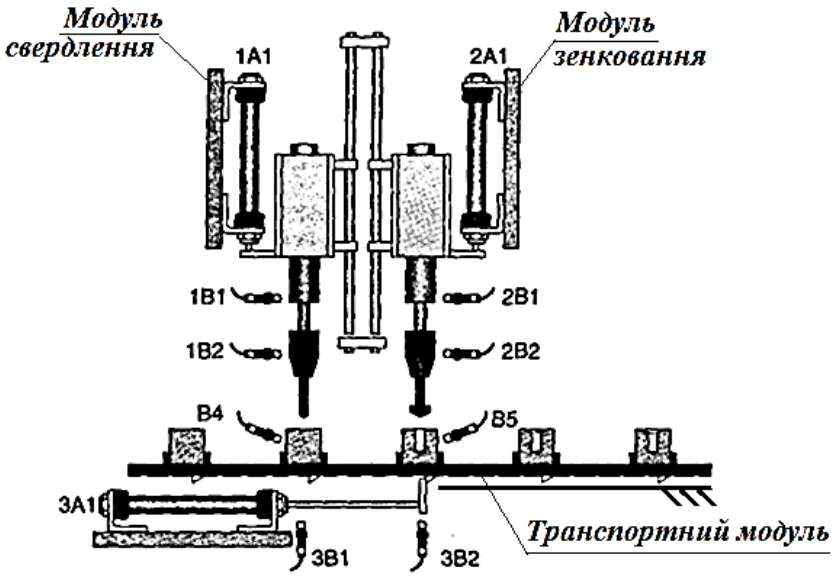


Рис. 2.11. Приклад застосування циклового програмного керування (ЦПК) в свердлильно-фрезерувальні верстаті

Висунення штока циліндра конвеєра фіксує датчик 3В2. Система не може гарантувати, що після кожного руху під свердлом і розгорткою завжди знаходитимуться деталі. Тому за відсутності наступної деталі відповідної прилад повинен бути зупинений. Коли одночасно відсутні обидві деталі, жоден з приладів не повинен включитися.

2.3.3. **Числове-програмне керування верстатами машинобудування (CNC-верстатами)**

Верстати з *програмним керуванням* по виду системи керування поділяють на верстати з системами циклового програмного керування (ЦПК) и верстати з системами **ч**ислового **п**рограмного **к**ерування (**ЧПК** – поширена назва **CNC**). В системах ЦПК програмується

цикл роботи верстату, а величини робочих переміщень, а саме геометрична інформація, задається спрощено, наприклад, за допомогою технологічних упоров (кінцевих перемикачів). У CNC-верстатах програмується і геометрична, і технологічна інформація.

CNC-система - це сукупність спеціалізованих пристроїв, методів і засобів, необхідних для реалізації CNC-верстатом своїх функцій, яка призначена для видачі керуючих команд виконавчим механізмам верстатів у відповідності з програмою керування і насамперед команд керування кроковим двигунам, вентильним двигунам або сервоприводу.

Числове програмне керування (**CNC**) — керування технічним об'єктом, при якому програму задають у вигляді масиву числової (дискретної) інформації. Керування технологічними циклами відбувається за допомогою програмованих мікроконтролерів або просто контролерів.

Структурна схема системи CNC-верстатом наведена на рис. 2.12. Інформація про кресленик деталі, яка підлягає механічної обробці на CNC-верстаті, одночасно надходить в модуль підготовки програми (*МПП*) і модуль технологічної підготовки (*МТП*). Останній забезпечує МПП даними про розроблений технологічний процес, режим різання та ін. На засадах даних розробляється програма керування. Наладчик встановлює на верстат пристосування, робочі інструменти у відповідності з документацією МТП. Встановлення заготовки та зняття обробленої деталі виконуються вручну оператором або виконуються автоматичним маніпулятором.

Функції *МПП* і *МТП* в сучасних CNC-верстатах виконує персональний комп'ютер на якому складається програма керування, яка потім переноситься в пам'ять мікроконтролера CNC-верстата. Цільові механізми з індивідуальним приводом на крокових двигунах (КД) реалізують основні і допоміжні рухи циклу обробки деталі. Датчики зворотного зв'язку (Дзз) на основі інформації про фактичне положення, переміщення, швидкість виконавчих механізмів, розмір обробленої поверхні, температурні та силові параметри технологічної системи контролюють фактичну величину координатних переміщень столу верстату, супорту, шпинделю при виконанні запрограмованого циклу роботи CNC-верстату.

Процес технологічної підготовки керуючих програм для верстатів з ЧПК за допомогою інтегрованих CAD/CAM-систем полягає з двох етапів:

- геометричне 3D-моделювання деталі в CAD-системі;

- вибір розмірів і матеріалу заготовки, вибір ріжучого інструменту і режимів обробки, автоматизоване проектування керуючої програми в G-кодах у CAM-системі з подальшою трансляцією через *постпроцесор* відповідного верстата.

CNC-система може видозмінюватися залежно від виду програмоносія, способу кодування інформації та методу її передачі в систему CNC. Пристрій CNC розміщують поруч з верстатом (в одному або двох шафах) або безпосередньо на верстаті (в підвісних або стаціонарних пультах керування). Двигуни приводів подач CNC-верстатів, що мають спеціальну конструкцію і є складовою частиною CNC-системи.

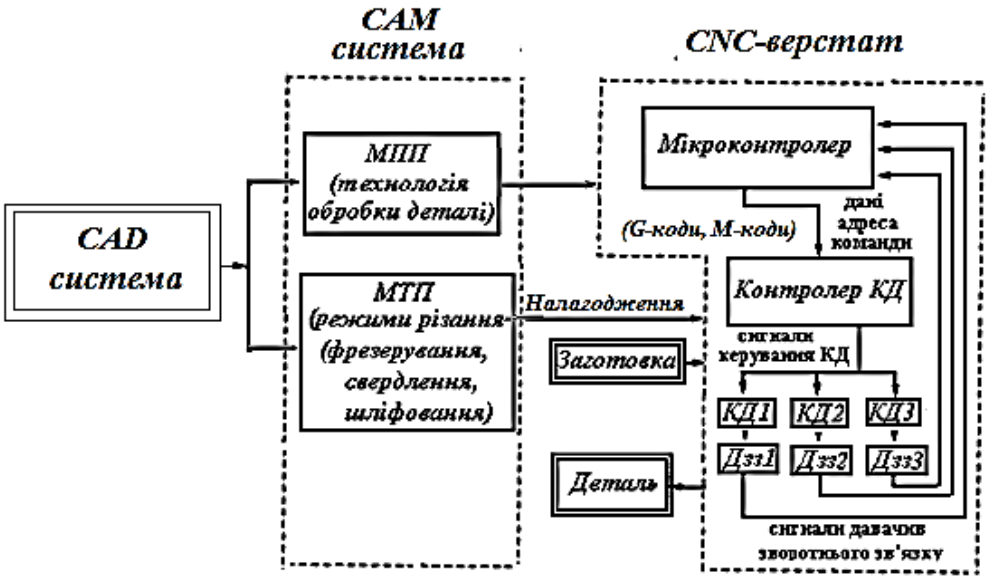


Рис. 2.12. Структурна схема CNC-верстата, де МПП - модуль підготовки G-кодів, МТП - модуль підготовки М-кодів

Всі дані, необхідних для обробки заготовки на верстаті містить два види інформації – *геометричну* і *технологічну*. Геометрична інформація - координати опорних точок траєкторії руху інструменту, а технологічна інформація - дані про швидкість, подачі, тип і номер ріжучого інструменту.

Найважливішою технічною характеристикою CNC-систем є її роздільна здатність або дискретність, а саме мінімально можлива величина лінійного і кутового переміщення робочого органу верстату, яка відповідає одному керуючому імпульсу, який постійно контролюється в процесі роботи CNC-машин і CNC-верстатів. Більшість сучасних CNC-систем мають дискретність 0,01 мм/імпульс і є приклади 0,001 мм/імпульс.

Програмовані контролери - це пристрої керування електроавтоматикою верстату або машини і побудовані за модульною архітектурою, що складається з мікроконтролера, пам'яті, портів вводу/виводу (*Input/Output*) та джерела живлення.

Принцип роботи контролера - *CPU* опитує порти вводу/виводу через загальну шину. Отримані дані про стан виконавчих механізмів (0/1) і датчиків аналізуються і порівнюються на відповідність запрограмованому стану цих механізмів і датчиків. При цьому процесор виконує арифметичні і логічні операції на даними і результат розрахунків видається на відповідний порт, до якого фізично підключено зовнішні виконавчі механізми і датчики металообробного верстату (токарного, фрезерного, шліфувального, та ін.) або технологічної машині певної галузі легкої промисловості (наприклад, швейної, трикотажної, взуттєвої та ін.).

В програмованих контролерах використовують *різни типи пам'яті* для зберігання програми дій з арифметичними і логічними операціями з даними про стан виконавчих механізмів (0/1) і датчиків металообробного верстату або технологічної машини:

- електричну енергонезалежну перепрограмовану пам'ять;
- оперативну пам'ять (ОЗУ) з вільним доступом;
- електрично перепрограмовану типу «флеш-пам'ять».

Програмований контролер має *систему діагностики* помилок при роботі технологічної машини:

- портів вводу/виводу;
- процесора;
- пам'яті;
- джерела живлення;
- шин внутрішніх і зовнішніх зв'язків.

Програмоносій може мати як геометричну, так і технологічну інформацію. Технологічна інформація забезпечує технологічний цикл роботи верстату, а геометрична інформація пов'язана з формою, розмірами оброблюємої заготовки і інструмента та їх взаємним положенням по трьом координатам.

Класифікація CNC-систем. Системи CNC класифікують за наступними ознаками:

- *по рівню технічних можливостей;*

по технологічному призначенню;

- *по числу потоків інформації* **:** незамкнені, замкнені, адаптивні (що самі пристосуються);
- *по принципу завдання програми* (в декодованому вигляді, тобто в абсолютних координатах або в приростеннях координат от ЕОМ);
- *по типу привода* (ступінчастий, регульований, слідкуючий, кроковий);
- *по числу одночасно керованих координат;*
- *по способу підготовки і вводу програми керування верстатом.*

По рівню технологічних можливостей за міжнародним стандартом системи CNC і CNC-верстати поділяються на наступні рівні:

NC* – системи з по кадровим читанням перфоленти на протязі циклу обробки кожній заготовки (* – попередні покоління верстатів);

SNC* – системи з однократним читанням перфоленти перед обробкою партії однакових заготовок;

CNC (**C**omputer **N**umerical **C**onrol) – **ЧПК** (системи з **Ч**исловим **П**рограмним **К**еруванням);

DNC – системи ЧПК групами верстатів від одного контролера (міні-ЕОМ);

HNC – оперативні системи з ручним набором програм в G-кодах і в М-кодах на пульте керування.

За технологічним призначенням CNC-системи поділяються на наступні чотири види:

CNC-системи позиційні (циклові);

CNC-системи, які забезпечують прямокутне формоутворення;

CNC-системи, які забезпечують прямолінійне формоутворення;

CNC-системи, які забезпечують криволінійне формоутворення.

Позиційні CNC-системи забезпечують високоточне координатне переміщення робочого органу (РО) верстату за програмою позицію за мінімальний час. По кожної координатної осі програмується тільки величина переміщення, а траєкторія переміщення може бути довільної. Переміщення РО між позиціями виконується з максимальною швидкістю,

а підхід до заданої позиції виконуються програмно з мінімальної («повзучої») швидкістю. Точність позиціювання підвищується в результаті підходу РО до заданої позиції завжди з одної сторони (наприклад, зліва направо). Позиційними системами CNC оснащають свердлильні та координато-розточувальні верстати.

CNC-системи*, що забезпечують прямокутне формоутворення*, на відміну від позиційних систем, дозволяють управляти переміщеннями РО верстату в процесі роботи. РО верстату програмно переміщується по координатним осям почергово, тому траєкторія інструменту має ступінчатий вигляд, а кожен елемент цієї траєкторії розташований паралельно координатним осях. Щоб скоротити час переміщення РО з однієї позиції в іншу, в ряді випадків використовують одночасний рух по двох координатах. При грубому позиціонуванні підхід РО до заданої позиції здійснюється з різних сторін, а при точному позиціонуванні - завжди з одного боку. Число керуючих координат в таких системах досягає 5, а число одночасно керованих координат - 4. Зазначеними системами оснащують токарні, фрезерні, розточувальні верстати.

CNC-системи*, що забезпечують прямолінійне формоутворення* (під любим кутом до координатних осей верстатів), управляють рухом інструменту при різанні одночасно за двома координатними осями (0X і 0Y). В таких системах використовують двокоординатний інтерполятор керуючих імпульсів відразу на два привода подач. Загальне число керуючих координат в таких системах 2…5. Зазначена система має широкі технологічні можливості (порівняно з прямокутними) і застосовуються для оснащення токарних, фрезерних, розточувальних і інших верстатів.

CNC-системи*, що забезпечують криволінійне формоутворення*, дозволяють управляти обробкою плоских і об'ємних деталей, що містять ділянку зі складних криволінійних контурами.

Контурні CNC-системи забезпечують прямокутне і криволінійне формоутворення.

Для розширення технологічних можливостей багатоцільові свердлильно-фрезерно-розточувальні верстати оснащують *контурне-позиційними* CNC-системами і вони переходять в клас CNC-верстатів.

По числу потоків інформації CNC-системи поділяються на розімкнені, замкнуті і адаптивні.

Розімкнені CNC-системи характеризуються наявністю одного потоку інформації від пристрою, що зчитує до робочого органу верстату. В

розімкненої системі нема датчика зворотного зв’язку і тому відсутня інформація про дійсне положення виконавчих органів верстату.

Замкнуті CNC-системи характеризуються двома потоками інформації - від пристрою, що зчитує і від датчика зворотного зв’язку. У цих системах неузгодженість між заданим і дійсним величинами переміщення виконавчих органів усувається завдяки наявності зворотного зв'язку.

Адаптивні CNC-системи - дозволяють коригувати програму обробки з урахуванням реальних умов різання і характеризуються наступними сигналами:

від пристрою, що зчитує інформацію;

від датчиків зворотного зв'язку;

від датчиків, встановлених на верстаті, які контролюють процес обробки за *такими параметрами*:

знос ріжучого інструменту;

зміна сил різання і сил тертя;

зміна твердості матеріалу заготовки, яка обробляється;

коливання і вібрації станіни із-за недостатньої жорсткості системи «верстат-заготовка».

2.3.4. **Інтегрування систем циклового мехатронного керування у середовище систем керування типу «розподільний вал»**

Об’єднання жорсткої системи керування типу «***розподільний вал***» і гнучкої система циклового керування технологічних машин легкої промисловості дозволяє зберегти переваги «*механіки*» і одночасно розширити інтелектуальні можливості «*заліза*».

На першому етапі проектування комп’ютерних технологічних машин (технологічних машин з мікропроцесорним керуванням) конструктору-розробнику (*designer-developer*) потрібно обирати і складати, як «пазл» узагальнену структуру технологічної машини з наступних складових: *технологічної* (**Т**) складової, яка пов’язана з функціональним призначенням машини; *механічної* (**М**) складової, яка пов’язана з кінематичною структурою механізмів машини та технологією машинобудування; *електромеханічної* (**ЕМ**) складової, яка пов’язана з типом приводу головного валу машини і з типом приводу виконавчих механізмів цільового призначення; *електронної* (**Е**) складової, яка пов’язана з типом контролера;

інформаційної (програмної) (**І**) складової, яка пов'язана з програмним забезпеченням і мовою програмування контролера. Перші дві складові є механіко-технологічної складової машин з мікропроцесорним керуванням а останні три складові можуть бути об'єднані в *енерго – інформаційну* складову (ЕнІ). Технологічна і механічна складові є обов'язкові для будови будь якої промислової або побутової технологічної машини галузі і проектування таких машин базуються на фундаментальних інженерних дисциплінах - ТММ, опір матеріалів, деталі машин, матеріалознавство та програмних продуктів CAE/CAD/CAM для автоматизованих розрахунків, моделювання та виготовлення позначених складових механіко-технологічної системи. Але технологічна машина при цьому утворюється з ручним керуванням, джерелом енергії для якої є мускульна сила людини, керування (*прямі зв'язки*) і контроль (*зворотні зв'язки*) за роботою таких технологічних машин залишаються також за людиною-оператором. Наприклад, побутові швейні машини з ручним/ножним приводом, плосков'язальні машини з переміщенням кареток вручну мають тільки технологічну і механічну складові у своєї структурі. При додаванні ЕМ-складової у вигляді електроприводу, гідроприводу або пневмоприводу технологічна машина перетворюється в машину з жорсткою системою керування типу «РВ». Додаванні електронної і інформаційної складових перетворює технологічну машину в машину з мікропроцесорним керуванням, у якій об'єднані жорстка система керування типу «*розподільний вал*» і гнучка циклова (мехатронна) система керування.

З урахуванням наведених складових доцільно надати наступні нові формулювання назв сучасних машин легкої промисловості.

В сучасної швейної машині загального призначення *механізм голки*, *механізм човника* і *механізм ниткопритягувача* залишаються із жорсткою системою керування типу «РВ» тому, що повинна бути високошвидкісна синхронізація робочих органів робочих органів наведених механізмів між собою, нитками і матеріалом, деталі з якого з'єднуються нитковим швом.

Наприклад, час взаємодії носика човна або петельника з петлею-напуском голкової нитки при 6000 обертах головного валу складає менше 1 мс. Тому тільки за допомогою механічної складової машини може бути така високошвидкісна чітка взаємодія в функції кута повороту головного валу металевих петле утворюючих робочих інструментів (голки, човника або петельника та ниткопритягувача або ниткоподатчика) з швейними нитками при петле утворюванні ниткових стібків. Така жорстка синхронізація

взаємодії петле утворюючих робочих органів швейної машини між собою, нитками і матеріалом з текстилю, трикотажу або шкіри закладається на стадії проектування циклограми роботи машини [15]. З цією проектної вимоги випливають жорсткі вимоги до точності, допускам на розміри деталей і посадкам для рухомих деталей, які утворюють кінематичні пари.

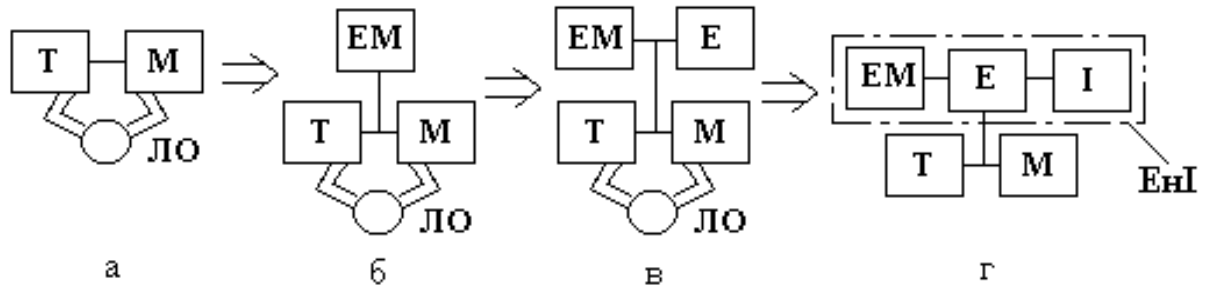


Рис. 2.13. Блок-схеми етапів інтегрування систем циклового мехатронного керування у середовище систем керування типу «розподільний вал»

На рис. 2.13 наведені існуючи і потенціальне можливі структурні компоновки технологічних машин легкої і текстильної промисловості з урахування технологічної, механічної і енерго-інформаційній (ЕнІ) складових механіко-технологічних систем легкої промисловості.

Наведеним на рис.2.13 етапам інтегрування складових технологічних машин відповідають наступні дискретно-логічні рівняння обраного класу об’єктів за теорією множин:

$$ТМ := X = Т \cap М = \{ x_i : x_i \in Т \cap x_i \in М \} \quad (2.10)$$

$$ТМ := X = Т \cap М \cap ЕМ = \{ x_i : x_i \in Т \cap x_i \in М \cap x_i \in ЕМ \} \quad (2.11)$$

$$ТМ := X = Т \cap М \cap ЕМ = \{ x_i : x_i \in Т \cap x_i \in М \cap x_i \in ЕМ \cap x_i \in Е \} \quad (2.12)$$

$$ТМ := X = Т \cap М \cap ЕМ = \{ x_i : x_i \in Т \cap x_i \in М \cap x_i \in ЕМ \cap x_i \in Е \cap x_i \in І \} \quad (2.14)$$

Автоматизована технологічна машина розглядається як множина (ТМ := Х), яка при інтегрування складаються з підмножин – відповідних складових Елементами x_i множини ТМ := Х є функціональні модулі відповідних складових технологічної машини: технологічної (Т),

механічної (М), електромеханічної (ЕМ), електронної (Е) та інформаційної (І). Перетини цих підмножин на діаграмі утворюють в площині універсальної множини, відповідно, дволистник, трилисник, чотирилистик та п'ятилисник з різними кількісними показниками різної вартості проектуванні і виготовлення модулів складових технологічної машини або умовною різною вагою значущості модулів складових технологічних машин галузі.

При проектуванні систем циклового програмного керування автоматизованими технологічними машинами доцільно використовувати наступні кроки алгоритму проектування:

1. Виконати декомпозицію механіко-технологічної системи на складові.
2. Обрати необхідні виконавчі механізми (електромагнітні, електромашинні, пневматичної або гідравлічної дії), датчики і ПЛК).
3. Скласти у відповідності з технічним завданням на проектування технологічний і функціональний графи.
4. Скласти систему рівнянь причинно-наслідкових зв'язків з урахування інтервалів часу затримки включення/вимкнення індивідуальних приводів виконавчих механізмів, а саме програмних установок параметрів таймеру/таймерів і рахівника/рахівників.
5. Скласти програму керування на одній з 5 мов міжнародного стандарту IEC 61131-3 на персональному комп'ютері.
6. Підключити виконавчі механізми і датчики до портів Input/Output (порти вводу/виводу) програмуючого логічного контролера (PLC).
7. Програму по п.5 компілювати і перенести в пам'ять PLC.
8. Виконати тестування роботи і налагодження пари *«технологічна машина (ТМ) - програмований логічний контролер (PLC)»*

Контрольні питання до розділу 2

1. Що визначає команди SET Y1, RESET YN1, RESET Y1 і команда SET YN1?
2. Чим відрізняється циклове БІстабільне керування приводом від циклового МОНОстабільного керування приводом ?

3. Чим відрізняється NC-машина від CNC-верстата ?

4. Що означає поширені скорочення **NC**, **SNC**, **CNC**, **DNC**, **HNC** ?

5. Які черговість етапів проектування циклового програмного керування технологічними машинами ?

6. Які зв’язки між етапами проектування циклового програмного керування технологічними машинами ?

7. Які різниця між електромеханічною машиною і машиною з електронним керуванням ?

8. Що таке енергетична і інформаційна складові технологічної машини ?

3. ОБ’ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ І СИНТЕЗ МЕХАТРОННИХ СИСТЕМ

3.1. **Вступ в об’єктно-орієнтоване проектування технологічних машин і верстатів машинобудування на засадах мехатроніки**

З появою нової ідеології (парадигми) програмування, з’явилися об’єктно-орієнтовані мови програмування і об'єктно-орієнтовані операційні системи. На їх базі створені сучасні об'єктно-орієнтовані САПР для твердотільного проектування (CAD-системи) і візуального проектування (Visual - системи). Перенос ідеології об'єктно-орієнтованого програмування [22]…[24] і основ побудови об'єктно-орієнтованих САПР на традиційні області проектування складних механіко-технологічних систем з комп’ютерним керуванням є не простим їх доповненням, а вимагає переосмислення задач проектування, як на стадії постанови проблеми проектування системи, так і на всіх наступних стадіях реального проектування, в тому числі і на завершальній стадії при виборі оптимального варіанту рішення. **О**б'єктно-**О**рієнтоване **П**роектування (**ООП**) або **О**б'єктно-**О**рієнтоване **М**оделювання (**ООМ**) - це автоматизоване (комп'ютерне) проектування (програмування) на засадах наступних основних принципів: *декомпозиції, інкапсуляції, поліморфізму, спадкування* і принцип *делегування*.

Об'єкт – це самостійний елемент (примірник) декомпозованої механіко-технологічної системи, модель якого описується ***п о л е м*** і ***м е т о д о м*** або *о б’ є к т* це сукупність інкапсульованих ***полів*** і ***методів.*** Кожний об’єкт відповідає за конкретні задачі, сформульованими в

термінах і поняттях для побудови програмного коду і послідуючого автоматизованого проектування на основі інформаційних технологій.

Команди для проектування системи (у програмних кодах) відбуваються за допомогою взаємодії ***об'єктів*** і по суті це є моделювання системи. Зміст основних етапів *об'єктно-орієнтованого проектування* (ООП), а саме о*б'єктно-орієнтованого аналізу* і *об'єктно-орієнтованого синтезу* базується на вихідні поняттях, що запозичені з *об'єктно-орієнтованого програмування,* в основу якого покладені наступні основні принципи розробки проектів сучасних комп'ютерних технологій:

принцип декомпозиції системи на класи і об'єкти**;**

***принцип інкапсуляції*;**

***принцип успадкування*;**

***принцип поліморфізму*;**

принцип делегування об'єкту обов'язків іншого об'єкта системи.

Принцип інкапсуляції це об'єднання в математичних моделях об'єктів різних базових класів, які мають різну фізичну природу і різні незалежні узагальнені координати. Діаграма об'єктів – *це графічне зображення інкапсуляції об'єктів.* ***Принцип успадкування*** об'єктів дозволяє різним об'єктам спільно використовувати одні і ті ж *методи* (алгоритми), а ***принцип поліморфізму*** **–** модифікувати методи для конкретних типів даних (полів).

При проектуванні об'єктно-орієнтованих механіко-технологічних систем «**механіка**» (рис. 1.2) успадковує необхідність створення базового класу *об'єкт-механізм* та базового класу *об'єкт-робочий інструмент* (рис. 3.2 і рис. 3.4), «**електроніка**» успадковує необхідність створення базового класу *об'єкт-мехатроніка* (рис. 3.6). Базові класи *об'єкт-механізм* та *об'єкт-робочий інструмент* на програмному рівні взаємодіють між собою та з базовим класом *об'єкт-сировина* та з базовим класом *об'єкт-робоче середовище.*

Потрібно розрізняти наступні дві топології проектування технологічних машин легкої промисловості або механіко-технологічних систем легкої промисловості:

1 – *топологію традиційного конструктивного проектування;*

2 – *топологію об'єктно-орієнтованого конструктивне-технологічного проектування на засадах мехатроніки і комп'ютерних технологій*.

При традиційному конструктивному проектуванні машин легкої промисловості після розробки проектної циклограми роботи машини виконується проектування цільових механізмів за класичними законами теорії механізмів і машин та законів теоретичної механіки. При цьому в технічну характеристику на машину завод-виробник закладає фізико-механічні властивості текстильних матеріалів, які обробляються технологічною машиною але які не закладені при проектуванні «заліза» і які є зовнішнім середовищем для робочих органів механізмів та не відносяться до машинобудування.

Зміст *традиційного конструктивного проектування* технологічних машин легкої і текстильної промисловості полягає в тому, щоби спроектувати цільові механізми машини, що розглядаються як самостійні елементи проектування. Для цього при традиційному конструктивному проектуванні спочатку виконується структурний аналіз та метричний (геометричний) синтез механізму цільового призначення, що пов'язані з визначенням працездатності механізму (вид і тип кінематичних пар) та з конструюванням ланок і механізмів машин (визначення довжин ланок, координат стояків). Потім при кінематичному аналізі визначаються функції положень, швидкостей і прискорень робочих інструментів. І далі при динамічному аналізі і синтезі виконується оптимізація розмірів і форми ланок, навантаження (реакції) в кінематичних парах і приведений до головного валу момент інерції механізмів, що пов'язані з конструюванням механізмів і машини. Наприклад, щоб спроектувати трикотажну машину достатньо спроектувати її цільові механізми: механізм ниткоподачи, механізм в'язання (механізм петле утворювання), механізм відбору голок, механізм товаро відводу та інші. Ці механізми можуть мати різну фізичну природу – механічну (для петлеутворення), електромеханічну (для приводу і засобів регулювання щільності в'язання), електромагнітну (для відбору голок програмованим селектором), електронну (для програмування рапорту, кольору фрагментів малюнку жакарду, інтарсії та інших переплетень), пневматичну (для товаро відводу полотна), оптичну (для контролю і сигналізації обриву ниток, пряжі) та інші. Правильні команди для цільових механізмів забезпечують нормальну роботу технологічної машини-автомату. Для цього повинна бути чітке узгодження роботи механізмів як між собою, так і з параметрами об'єкта обробки на всій трасі ниток (пряжі) до готового продукту (трикотажного полотна або трикотажного виробу). Тому

конструктивне проектування починається з проектування циклограми роботи машини. При конструктивному проектуванні в'язальних машин (швейних машин, ткацьких верстатів) властивості об'єкта обробки (нитки або ниток) враховуються, як зовнішні фактори по відношенню до технологічної машини і ці фактори залишаються незмінними на час проектування машини. Топологія традиційного конструктивного проектування швейних машин наведена на рис. 3.1, в'язальних машин – на рис. 3.3 і мехатронних систем технологічних машин – на рис. 3.5.

Зміст *об'єктно-орієнтованого конструктивне-технологічного проектування* технологічних машин легкої і текстильної промисловості полягає в тому, щоб спроектувати *механіко-технологічну систему* з *мехатронною системою керування*. При такому проектуванні, що базується на об'єктно-орієнтованому підході з використання сучасних інформаційних технологій проектування технологічних машин галузей швейного, трикотажного і взуттєвого виробництв потрібно розглядати, як проектування робочого процесу, а саме врахування базового класу об'єкт-сировина з підкласами об'єкт-нитки, об'єкт-тканина, об'єкт-трикотаж, які є складовими проектування швейних і в'язальних машин (рис. 3.2 і рис. 3.4). При застосування шкіри і шкіроподібних матеріалів для отримання 3D-форми заготовок взуття на обтягувально-затягувальних машинах базовий клас об'єкт-сировина містить підклас об'єкт-шкіра (рис. 3.2).

Цільовою функцією об'єктно-орієнтованого конструктивне-технологічного проектування є не функція положення, наприклад, робочого органу машини, як при конструктивному проектуванні, а цільовою функцією *є функціональне призначення кінцевого продукту* – результату робочого процесу, а саме ниткового шва у виробі виготовленому на швейної машині, тип переплетення та/або форма текстильних виробів, вироблених на в'язальних (трикотажних) машинах, форма деталей, складальних одиниць при виготовленні взуття.

Якому же методу проектування віддати перевагу ? Методи *традиційного конструктивного проектування* (рис. 3.1, рис. 3.3 та рис. 3.5), ціллю яких є цільове проектування механічних систем технологічних машин розроблені краще і часто залишаються традиційними. Але з появою об'єктно-орієнтованому підходу до проектування складних механіко-технологічних систем можна очікувати продовження розробок і широкого застосування методів *об'єктно-орієнтованого конструктивне-*

технологічного проектування з використанням сучасних комп'ютерних 3D-технологій моделювання та *мехатроніки*.

На рис. 3.1 наведена схема топології традиційного конструктивного проектування неавтоматизованих швейних машин.

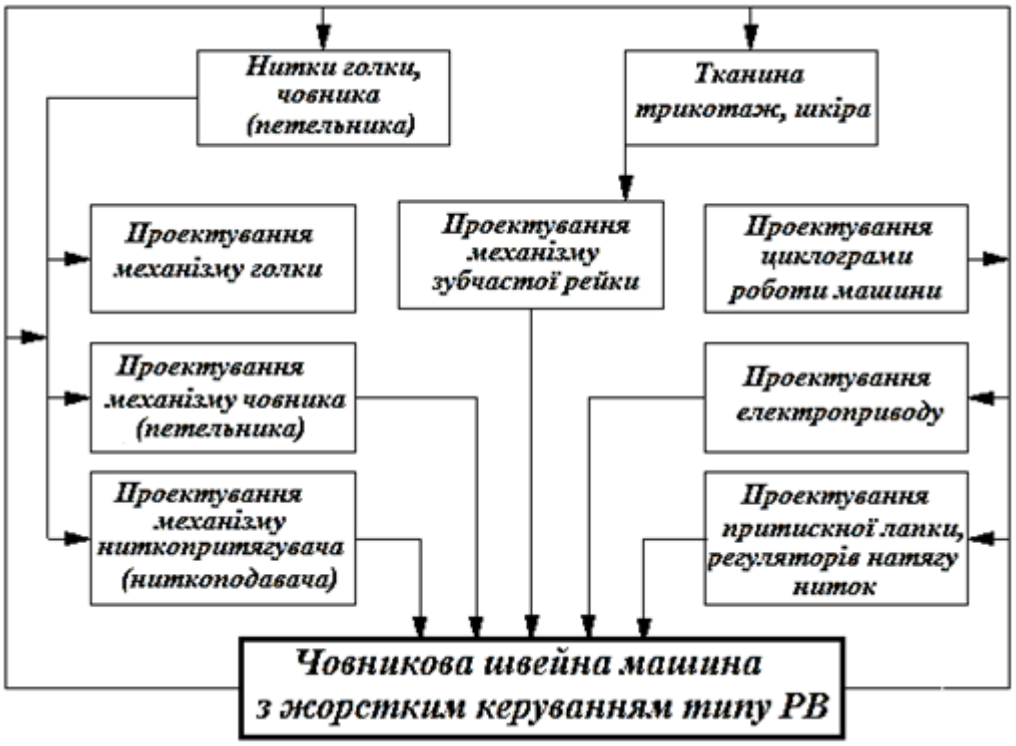


Рис. 3.1. Топологія традиційного конструктивного проектування неавтоматизованих швейних машин

Існуючі САПР-одягу, САПР-трикотажу, САПР-взуття не містять базових класів об'єкт-механізм, об'єкт-робочий інструмент, об'єкт-мехатроніка.

Для об'єктно-орієнтованого проектування робочих процесів CNC-машин (машин з ЧПК - числовим програмним керуванням) широко використовуються CAM-технології у вигляді програмних продуктів типу **ArtCAM**, **DelCAM**, **SolidCAM** та інших систем 3D-моделювання, які за допомогою постпроцесорів цільового призначення враховують кінематичні особливості конкретного CNC-верстату фірм виробників світового машинобудування. Самі верстати також проектуються з використанням сучасних технологій 3D-моделювання.

Це стосується і сучасного світового машинобудування легкої промисловості при проектуванні багатоголкових вишивальних автоматів,

автоматизованих швейних і в'язальних машин, розкрійних агрегатів з ЧПК та ін.

На рис. 3.2 наведена схема топології об'єктно-орієнтованого конструктивне-технологічного проектуванням швейних машин, як механіко-технологічної системи з застосуванням засобів мехатроніки.

Що стосується об'єктно-орієнтованого конструктивне-технологічного проектування мехатронних модулів (рис. 3.6), то етапи традиційного конструктивного проектування (рис. 3.5) зберігаються і переходять в стадію об'єктно-орієнтованого аналізу (ООА) - *перша стадія проектування об'єктно-орієнтованого конструктивне-технологічного проектування мехатронних систем.*. Розробка і налагодження програми керування на засадах рівнянь причино-наслідкових зв'язків це стадія об'єктно-орієнтованого синтезу (ООС) - *друга стадія об'єктно-орієнтованого конструктивне-технологічного проектування мехатронних систем.* При цьому на стадії ООС виконується оптимізація режимів технологічного процесу і/або оптимізація кінематики і динаміки технологічної машини.

ООА передбачає математичний і алгоритмічний опис закономірностей робочого процесу механіко-технологічної системи, а **ООС** – реалізацію алгоритму об'єкту проектування у програмному коді.

В класичному визначенні і в застосуванні до робочих процесів на основі механічної технології відповідних технологічних машин легкої і текстильної промисловості основні принципи ООП мають наступний зміст і застосування.

В роботі [25] на основі об'єктно-орієнтованого підходу розроблені наукові основ проектування складних механіко-технологічних систем в текстильній промисловості на прикладі автоматизованого проектування робочих процесів трикотажних машин. Для геометричних моделей різних структур трикотажу, автором запропанована узагальнена незалежна координата (ділянка нитки з постійними властивостями) для базового *об'єкту–нитки*. Для *об'єкту–нитки* розроблена оригінальне D-кодування,

Рис. 3.2. Топологія об’єктно-орієнтованого конструктивне-технологічного проектуванням автоматизованих швейних машин, як механіко-технологічної системи із застосуванням засобів мехатроніки

яке широко проілюстроване і узагальнене на конкретних прикладах механічної технології утворення різних трикотажних переплетень, переплетень ниток **основи** і **утка** при утворенні тканин і переплетень голкової і човникової ниток при утворенні машинних зигзаг стібків. При цьому показано місце D-кодування серед геометричних моделей в системах кодування трикотажу в САПР трикотажу. Схема топології традиційного конструктивного проектування в’язальних машин наведена на рис. 3.3.

Схема топології об’єктно-орієнтованого конструктивне-технологічного проектуванням в’язальних машин, як механіко-технологічної системи наведена на рис. 3.4.

На рис. 3.5 наведена схема топології традиційного конструктивного проектуванням мехатронних систем технологічних машин.

Рис. 3.3. Топологія традиційного конструктивного проектування в’язальних машин

Рис. 3.4. Топологія об’єктно-орієнтованого конструктивне-технологічного проектуванням в’язальних машин, як механіко-технологічної системи

Рис. 3.5. Топологія традиційного конструктивного проектуванням мехатронних систем технологічних машин

Для негеометричних моделей, а саме робочих процесів трикотажних машин в якості узагальненої координати запропонована іншу типову узагальнена координата – реакцію або відклик типового робочого інструмента – трикотажної голки за визначеною структурою проектованого трикотажу, послідовність наступних елементарних команд для трикотажних ***об'єкт-голка***: **П**–нитка (команда прокладання нитки на голки), **С**–нитка (команда скидання нитки), **О**–нитка (команда обрив нитки). Ці команди та інші команди ***об'єктів*** у англомовному запису стають ***методами*** своїх об'єктів, а потім і ідентифікаторами методів в блоках програми, записаний в відповідному коді. На основі об'єктно-орієнтованого аналізу виконане аналітичне уявлення емпіричних залежностей, де на відміну від загальноприйнятого уявлення вихідних даних, формул і емпіричних залежностей, вони представлені, як сукупність «***об'єктів-примітивів***» (крапки, експериментальних даних, скалярної функції, полінома) нефізичної природи зі своїми ***полями*** і ***методами***.

На рис. 3.6 наведена схема топології об’єктно-орієнтованого конструктивне-технологічного проектування мехатронних систем.

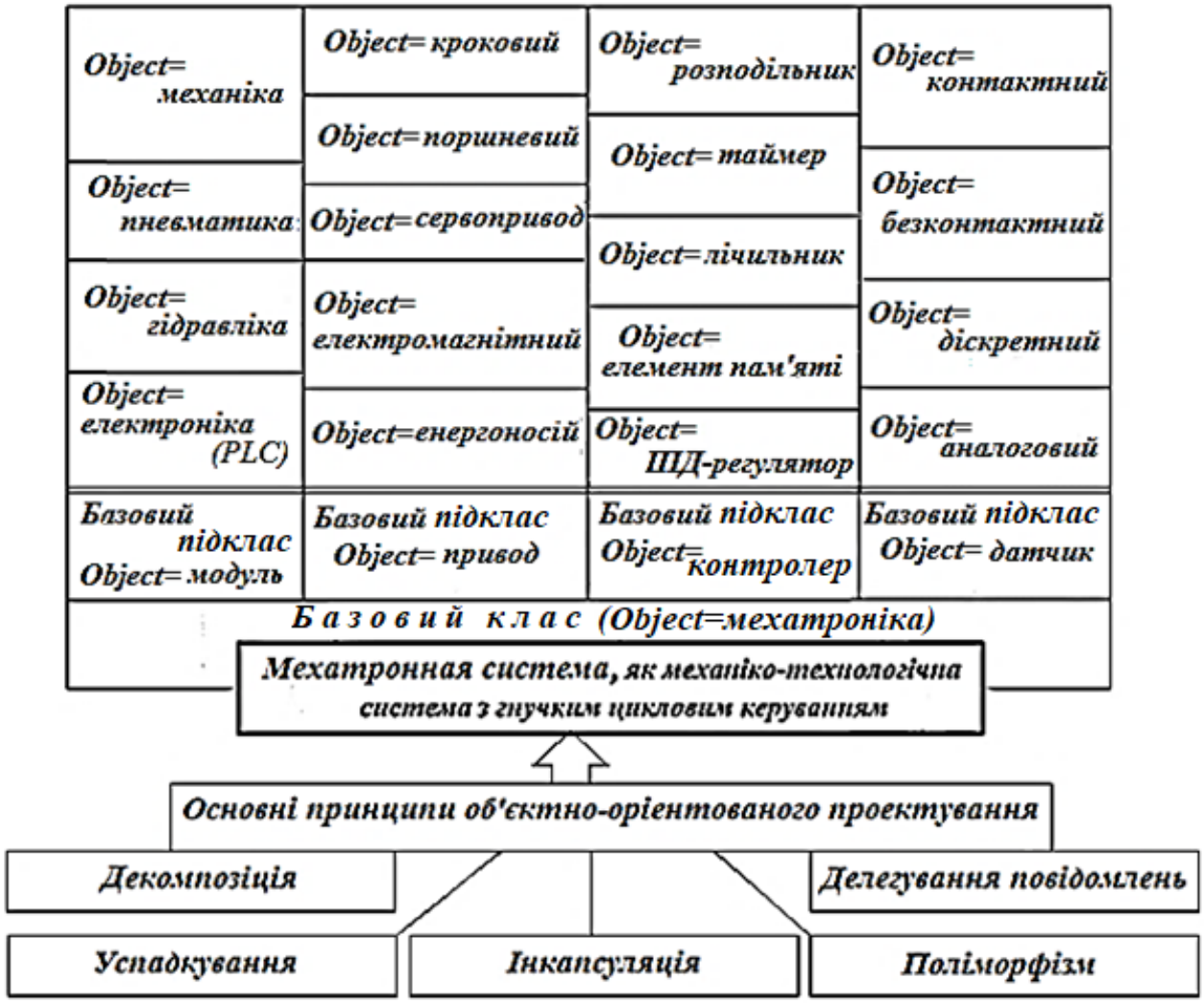


Рис. 3.6. Топологія об’єктно-орієнтованого конструктивне-технологічного проектування мехатронних систем

Основні поняття об’єктно-орієнтованого проектування механіко-технологічних систем на засадах мехатроніки

Клас – це безліч ***об’єктів*** цільового призначення з загальною структурою і поведінкою, які мають або не мають загального предку.

В базовому класі технологічних машин легкої промисловості, які реалізують механічну технологію обробки матеріалів легкої і текстильної промисловості, нащадками загального предку для швейних, взуттєвих і

текстильних машин є ***об'єкт-механізм***, ***об'єкт-ланка*** і ***об'єкт-робочий інструмент***. Для загального предку в базовому класі напівфабрикатів (сировини) нащадками-об'єктами з обліком принципу спадковості є ***об'єкт–нитка***, ***об'єкт-тканина*** (трикотаж, стібки, стрічки, шви), ***об'єкт-шкіра*** (шкіроподібні матеріали або деталі із них). В базовий клас апаратних і програмних засобів управління автоматизованої технологічної машини у механіко-технологічну систему входять ***об'єкт–датчик***, ***об'єкт-виконуючий механізм***, ***об'єкт–контролер*** та інші технічні засоби мехатроніки. Наведені базові класи об'єктів різної фізичної природи, різного принципу дії і з різними функціональними обов'язками не мають загального предка для цільових робочих процесів, як надкласу з точки зору ООП.

Об'єкт – це самостійний елемент проектованої системи, якій отриманий при *декомпозиції* складних механіко-технологічних систем, в тому числі мехатронних систем галузевого машинобудування цільового призначення, при цьому *об'єкт* має певні функціональні ***обов'язки***, які від ***делегує*** іншому об'єкту, якій з ним пов'язаний кінематичними, технологічними, структурними або інформаційними зв'язками.

Сучасні технологічні машини легкої промисловості це мехатронні механіко-технологічні системи, які мають чотири основних складові: ***механічну***, ***енергетичну***, ***інформаційну*** і ***технологічну***. При проектуванні мехатронних циклових систем більше уваги приділяється першим трьом складовим, а цільове технологічне призначення таких систем залишається поза увагою. Задачею метричного синтезу машин є як раз визначення потрібних технологічних переміщень робочих органів для функціонального призначення технологічної машини або пристрою до машини з метою автоматизації допоміжних або основних циклових робіт на засадах мехатроніки.

ООП формально не виключаючи основні стадії традиційного проектування, на яких відробляється конструкторська документація, дозволяє автоматизувати моделювання і вибір раціональних або оптимальних варіантів складних механіко-технологічних систем. Для цього необхідна розробка математичних моделей і алгоритмів проектування, функціонально і програмно зв'язаних з об'єктно-орієнтованим аналізом (ООА) і об'єктно-орієнтованим синтезом (ООС) об'єкту дослідження та з автоматизованим проектуванням.

ООА зв’язаний з побудовою математичних моделей (геометричних, логічних, структурних, емпіричних, алгоритмічних) на основі проектування діаграми класів, діаграми об’єктів, діаграми взаємодії об’єктів і інших графічних діаграм, відомих як «нотація Буча» [22]. Проте, “нотація Буча” вимагає уточнення в додатках до об’єктно-орієнтованого аналізу особливостей механіко-технологічних систем легкої промисловості.

Для побудови моделей, робочого процесу технологічної машини, основними обов’язками абстрактного (віртуального) ***об’єкту=робочий інструмент*** є подальша його використання і тому потрібна ініціалізація основних його дій, фіксуючих **полем** *об’єкту=робочий інструмент*, а саме питанням стану - «*увімкнений*», «*вимкнутий*»? Відповідь на це питання дають дії, які зв’язані з методом об’єкту, а саме наступні чотири дії, які зв’язані з реалізацією **методу** для цього об’єкту: *конструктор* **(Init),** *деструктор* **(Destr**), *встановити* **(Set**), *отримати* **(Get**), тобто визначення в активному або пасивному стані знаходиться *об’єкт=робочий інструмент* машини. Наприклад, надаємо абстрактному робочому інструменту ідентифікатор **Tool=object.** Тоді узагальнена модель для такого об’єкту=робочий інструмент маймо наступну типову структуру і вигляді фрагменту програми:

Об′єкт:

Tool=object ; {об′єктний тип}

Поле:

Active: Boolean ; {логічна умова стану об′єкта}

Методи:

Init; {конструктор об′єкту, який – ініціалізує вихідні дані про *об′єкт=робочий інструмент*}

Destr; {деструктор об′єкту - автоматично викликає команду, яка звільняє пам’ять, виділену під об’єкт і підтверджує виконання (**done** - зроблено). В рядку програми деструктор вказується у вигляді сполучення знаку „тильда” ~ за яким йде ім′я класу}

SetState; {встановлює стан увімкнений/вимкнутий або активен/пасивен робочий інструмент технологічної машини}

InsTF: Boolean; {*скорочення* від **I**nstall **T**rue or **F**alse - інсталювати логічний стан робочого інструмента: якщо InsTF=**T**rue (істина), то робочий

орган увімкнений (активний стан) і може виконувати вхідні команди у відповідності з діаграмою взаємодії об′єктів інакше InsTF=**F**alse і це означає, що або обірвана нитка, або пошкоджена голка, або машина не працює}

Наприклад, *полем* для абстрактного ***об'єкт=ланка*** є постійні вихідні дані такі, як геометричні розміри ланки та її інерційні і пружно-дисипативні параметри. Для ***об'єкт=нитка*** базового класу ***об'єкт=сировина*** (рис.3.2 і рис.3.4) *полем* є текс, номер нитки, міцність, крутка, число складень, рядок D-коду. *Полем* для абстрактного ***об'єкт=тканина*** одягу є модуль пружності матеріалу, жорсткості тканини при згині, рапорт та інші.

Для об'єктів базового класу ***об'єкт=контролер*** класу *полями* є тип і координати розташування виконаних механізмів, координати розташування датчиків крайніх положень робочих інструментів машини. **Методом** є повідомлення про стан цих об'єктів «увімкнений»/«вимкнутий». Запит відбувається при опитуванні вхідних портів (**Input**) і вихідних портів (**Output**) програмуючого контролера.

Сам контролер при ООП механіко-технологічної системи у програмному середовищі задається як підклас ***об'єкт=контролер*** базового класу ***об'єкт=мехатроніка*** (рис. 3.6). А саме як елемент агрегату на основі одного із принципів агрегації при побудові ієрархії складних систем абстрактних об'єктів. Для різних об'єктів вимагається формалізація описування своїх ***полів*** і ***методів***. Об'єктно-орієнтований аналіз доцільно починати з побудови цільових графічних діаграм, які далі використовуються для проектування моделей об'єктів базового класу об'єкт=технологічна машина.

Таким чином, з відомими (ініціалізованими) ***аргументами*** – *полем* або вхідними даними кожного об'єкту працює свій (також ініціалізований) ***метод***, який й є реакцією на повідомлення одержувача і який необхідний для задоволення прийнятого запиту. Отже, при ОБ'ЄКТНО-ОРІЄНТОВАНОМУ ПРОЕКТУВАННІ ***(моделюванні)*** робочих процесів механіко-технологічних систем основними примітивами є **ОБ'ЄКТ, ПОЛЕ** (аргументу або вхідні дані) і **МЕТОД** (спосіб реалізації *повідомлення* про *обов'язки* об'єкту, який прийняв *запит*). Тут ***запит, повідомлення, обов'язки*** це формальні поняття.

Далі необхідно формалізувати описування тих функцій, які механіко-технологічна система делегує об’єктам різних базових класів, наприклад, базового класу «робочі інструменти» і базового класу «текстильні матеріали», тобто описати їх формальні обов’язки, які піддаються алгоритмізації з точки зору ООА і графічно відобразити ці обов’язки об’єктів за допомогою діаграм.

3.2. **Основні принципи об’єктно-орієнтованого проектування**

Як було підкреслено основними принципами об'єктно-орієнтованого проектування (програмування) є «***принцип декомпозиції***», «***принцип інкапсуляції***», «***принцип успадкування***» (*рос*. принцип наследования), «***принцип поліморфізму***» і «***принцип делегування***», які застосовуються для створення алгоритмічних і комп'ютерних моделей на засадах інформаційних технологій.

Принцип декомпозиції – це один з принципів об'єктно-орієнтованого проектування (ООП) , згідно з яким складна механіко-технологічна система або систему іншої природи поділяється на ***об'єкти***. При цьому ***об'єкт*** це самостійний елемент проектуємої системи з певними функціональними обв'язками. Об'єкт уявляє собою сукупність у програмному змісті *інкапсульованих* **полів** і **методів** з точки зору ОПП

Принцип інкапсуляції – це об'єднання у програмному середовищі в одно ціле *поля* (даних про об’єкт) і *методу* (алгоритмів обробки цих даних). При інкапсуляції поля, як би вкладені в свій метод і, цей принцип ОПП дозволяє при автоматизованому проектуванні перебудовувати окремі блоки програмного коду, не торкаючи весь базовий клас об'єктів.

Принцип поліморфізму – це властивість програмних об'єктів-нащадків, які мають програмного об'єкта-предку *застосовувати різні методи об'єкта для вирішення одній і тієї задачі ООП*. Об'єкт, якій позначений поліморфним ім'ям (змінної), може бути об'єктом інших базових класів і може по своєму реагувати на один і той же набір операцій.

Принцип спадкування (поширена назва "Принцип наследования")

–це властивість програмних класів і програмних об'єктів породжувати своїх програмних нащадків. При одиночному спадкуванні один клас або об'єкт застосовує структуру та поведінку іншого класу та об'єкту, а при кількісному спадкуванні - застосовує структуру та поведінку інших класів та об'єктів.

Принцип делегування – це делегування об'єкту повноважень іншого об'єкту системи, коли один об’єкт, відповідальний за операцію, передоручає виконання цієї операції або реалізацію своєї поведінки іншому об’єкту. Такий альтернативний підхід до організації класів, без класів, дозволяє домогтися спільності поведінки об’єктів в системі без принципу спадкоємства. Якщо проектування складної системи засноване на делегуванні повноважень механіко-технологічної системи безпосередньо об’єктам, тоді класів немає. Діаграма класів – *графічне зображення структури генеалогічного дерева (графа) базових класів об’єктів.*

О*б'єктно-орієнтоване проектування* це сучасна комп’ютерна ідеологія проектування складних механіко-технологічних систем, яка запозичена з ідеології *об’єктно-орієнтованого програмування* алгоритмічних мов на засадах наступних переходів: ***процедури → модулі → абстрактні типи даних → об’єкти.***

Процедури –дозволяють сконцентрувати водному місці роботу (записати програмний код), яка виконується багатократно, а потім багатократно використовувати цей програмний код (шаблон), замість того щоб його писати знову і знову (*програмний код* це реалізація алгоритму програми на одної з алгоритмічних мов програмування).

Модулі (програмні) – дозволяють розділити *дані* (*поля*) і *методи* (*алгоритми*) на дві частини: відкриті (*public*), які доступні тільки ззовні модуля і закриті (*private*), які доступні тільки всередині модуля. Програмні модулі рішають проблему маскування інформації, а значить її цілісності.

Абстрактні типи даних – дозволяють генерувати декілька екземплярів абстрактного типу даних.

На прикладі систем із жорсткою системою керування типу «розподільний вал» і систем із цикловою системою керування, що програмується розглянемо об’єктно - орієнтовані обов’язки елементів, які отримані після їх декомпозиції механіко-технологічної системи.

3.3. **Об’єктно-орієнтовані обов’язки робочих органів машин при утворенні човникових стібків класу 300**

Для виконання човникового стібка в кожній швейній машині є наступні типові робочі органи: *голка, човник, ниткопритягувач, зубчата рейка*, притискна лапка, голкова пластина, регулятор натягу голкової нитки, регулятор натягу човникової нитки, регулятор довжини стібка і важіль реверсу матеріалу. Перші чотири робочих органів відносяться до стібкоутворюючих і розглядаються при побудові циклограми роботи машини, тому що їх положення залежить від кута повороту головного валу. Інші забезпечують якість утворення човникових стібків і швів і регулювання їх параметрів, а їх положення не залежить від кута повороту головного валу машини, тому що відсутні кінематичні зв’язки з головним валом.

Об’єктно-орієнтованими обов’язками голки є:

- проколювання матеріалів;
- проведення крізь матеріали голкової нитки;
- утворення петлі-напуску;
- *д е л е г у в а н н я* (передача) згідно з циклограмою роботи машини виконаних обов’язків човнику, ниткопритягувачу і зубчатої рейці.

Об’єктно-орієнтованими обов’язками човника є:

- захоплення петлі-напуску;
- розширення петлі-напуску;
- обведення петлі-напуску навколо шпуле тримача зі шпулькою з човниковою ниткою і шпульним ковпачком;
- утримання хвостовиком накладної пластини скинутої з носика човника петлі;
- переплетення голкової і човникової ниток для утворення вузлика;
- *д е л е г у в а н н я* (передача) згідно з циклограмою роботи машини виконаних обов’язків ниткопритягувачу і зубчатої рейці.

Об’єктно-орієнтованими обов’язками ниткопритягувача є:

- подача нитки голці при її русі в матеріалі до моменту утворення петлі-напуску;
- подача нитки човнику від моменту захоплення носиком петлі-напуску до моменту максимального розширення петлі-напуску;

- вибирання голкової нитки із човникового пристрою і змотування човникової нитки зі шпульки;

- затягування стібка і змотування голкової нитку з бобіни або котушки;

- *д е л е г у в а н н я* (передача) згідно з циклограмою роботи машини виконаних обов'язків зубчатої рейці.

Об'єктно-орієнтованими обов'язками зубчатої рейки є:

- переміщення матеріалу на задану довжину стібка із зупинкою при знаходженні голки в матеріалі;

- регулювання довжини стібка;

- реверс матеріалу для виконання закріпок на початку і в кінці шва;

- *д е л е г у в а н н я* (передача) згідно з циклограмою роботи машини виконаних обов'язків голці, притискної лапки голкової пластини.

Об'єктно-орієнтованими обов'язками притискної лапки є:

- забезпечення кінематичного зв'язку в системі: зубчата рейка – матеріал – підошва притискної лапки при переміщенні матеріалу на задану довжину стібка і в системі: зубчата рейка – матеріал – голкова пластина при вистою матеріалу;

- *д е л е г у в а н н я* (передача) функцій педалі фрикціону і головному валу на початок процесу шиття при знаходженні притискної лапки в нижньому положенні;

- втиснення мікро об'єму текстильного матеріалу до зубчатої рейки на фазі переміщення (волочіння) матеріалу при знаходженні голки над матеріалом;

- *д е л е г у в а н н я* (передача) функцій головному валу, а значить всім стібкоутворюючим робочим органам на закінчення процесу шиття при знаходженні притискної лапки в верхньому положенні.

Об'єктно-орієнтованими обов'язками голкової пластини є:

- утримання матеріалу під притискною лапкою при знаходженні голки над матеріалом і в матеріалі і наявності в голкової пластині круглого отвору, скрізь центр якого проходить вісь голки;

- розділення траєкторії вершин зубців зубчатої рейки на робочу і холосту за рахунок наявності в голкової пластині подовжніх прорізів для проходження зубчатої рейки;

- *д е л е г у в а н н я* (передача) притискної лапки і зубчатої рейки необхідної умови переміщення матеріалу при знаходженні голки над матеріалом і умови вистою матеріалу при знаходженні голки в матеріалі.

Об'єктно-орієнтованими обов'язками регулятора натягу голкової нитки, регулятору натягу човникової нитки, регулятору довжини стібка і важеля реверсу матеріалу випливають з їх назви.

Розглядаючи процес утворення човникового стібка на різних швейних машинах, можна помітити, що більша частина операцій, що виконуються робочими органами, є подібними, а способи утворення стібка можуть бути різними.

Переплетення ниток при утворенні човникового стібка виконується за допомогою коливного або обертового човника.

В швейному, трикотажному і взуттєвому виробництвах переважає механічна технологія утворення стібків, яка в узагальненому вигляді при системному аналізі має на вході три або чотири «інгредієнта» - швейні нитки, швейну машину і текстильний матеріал або швейні нитки, фурнітуру, швейну машину і текстильний матеріал. На виході механічної технології утворення стібків - результат реалізації механічної технології. З кожним виконаним машинним циклом зменшується різниця між «інгредієнтами» і результатом механічної технології. І коли ця різниця дорівнюється нулю, тоді отримується новий технологічний об'єкт, з новими фізико-механічними властивостями, новим призначенням і новими експлуатаційними властивостями.

3.4. Об'єктно-орієнтовані обов'язки елементів мехатронного модуля

Розглянемо об'єктно-орієнтовані обов'язки елементів циклової системи керування на засадах МОНОстабільного керування пневмоциліндром на прикладі об'єктно-орієнтованих обов'язків типової пари елементів мехатроніки «пнемо- , гідроциліндр» – «пнемо, - гідро розподільник»:

Об'єктно-орієнтовані обов'язки пневмоциліндру *(гідроциліндру)* – перемножити тиск p у без штоковій камері на площу S поршня і *делегувати* результат в вигляді сили $F = p \cdot S$ на шток поршню.

Об'єктно орієнтовані обов'язки золотника пневморозподільника – отримати рух від малої електромагнітної сили руху якоря електромагніту і перетворити цю силу в значущу силу $F = p \cdot S$ руху поршню.

Об'єктно орієнтовані обов'язки пневморозподільника – отримати команду від електромагніту і виконати рух поршня, в результаті чого клапан *делегує* ***пневмоциліндру*** *(гідроциліндру)* знак множення для виразу $F = p \cdot S$. Якщо клапан включений – тиск і площа перемножуються, якщо клапан вимкнений – добуток дорівнюється нулю.

Об'єктно-орієнтовані обов'язки електромагніту пневморозподільника – сприйняти команду на включення/вимкнення від кінцевих вимикачів або від кнопки «Start» на вході для множення магнітної постійної $\mu_0 \left[\frac{\text{Гн}}{\text{м}} = \frac{\text{кг}\cdot\text{м}^2}{\text{с}^2\cdot\text{А}^2\cdot\text{м}}\right]$, квадрату сили струми $\boldsymbol{I}$ $[\text{А}^2]$, квадрату кількісті витків $\boldsymbol{w}$ котушки, квадрату площі $S^2[\text{м}^2]$ перетину сердечника і ділення результату на $2l_c^2$, де $l_\text{с}$ – середня довжина повітряного зазору. Отриманий результат *делегується* електромагніту=object у вигляді електромагнітної сили $F_e = (\mu_0 I^2 \cdot w^2 S^2)/2l_c^2$ для переміщення золотника пневморозподільника.

Об'єктно-орієнтовані обов'язки штоку поршня – включити сигнал стану (при необхідності шток забезпечує кінематичний зв'язок з передаточним механізмом) і *делегувати* результат за допомогою кінцевого вимикача (геркона або іншого датчика положення) контролеру і веденої ланці передаточному механізму.

3.5. **Приклад об'єктно-орієнтованого аналізу технологічної машини**

Послідовність команд робочого процесу машини визначається діаграмою (циклограмою, синхрограмою або в загальному випадку знаковою моделлю) взаємодії робочих інструментів між собою, нитками і матеріалом за 1 цикл утворення стібка. На рис. 3.7 на прикладі технологічної машини (швейної машини загального призначення човникового стібка) наведена об'єктно-орієнтована діаграма взаємодії об'єктів двох базових класів:

базового класу об'єкт=робочі інструменти ***(Tool=object)***;
базового класу об'єкт=сировина ***(RawMaterial=object).***

Ці базові класи не мають загального предка. Базовий клас об'єкт=контролер на діаграмі взаємодії об'єктів не зображений.

Діаграма взаємодії об'єктів це графічне зображення перетинання базових класів об'єктів, які не мають або мають загального предка. При цьому перетини об'єктів обумовлені функціональним призначенням машини і порядком їх взаємодії за один оборот головного валу машини. Тобто така взаємодія об'єктів визначається функціональним призначенням машини і порядком взаємодії «*об'єт*=робочі інструменти» і «*об'єкт* =напівфабрикати» між собою за один цикл виконання одного стібка. По аналогії будуються діаграма взаємодії об'єктів і для інших технологічних машин (для розкрою, пошиття, в'язання, формування), які реалізують механічну технологію виготовлення одягу, взуття і інших виробів або напівфабрикатів із матеріалів легкої і текстильної промисловості.

Сформулюємо основні обов'язки, які підлягають формалізації об'єктів трьох базових класів **Tool=object**, **RawMaterial=object** і **Link=object**.

Формалізованими об'єктно-орієнтованими обов'язками делегованими механіко-технологічною системою для базового класу Object=робочі інструменти (рис. 3.2) або **Tool=object** {об'єктний тип} є наступні **вказівки-команди**:

1. *Object=голка* або **Needle=object –** провести голкову нитку (об'єкт суміжного базового класу) скрізь пакет зшиваємих текстильних матеріалів і утворити петлю-напуск.

2. *Object=човник* або **Chuttle=object** – захопити петлю-напуск, переплести між собою голкову нитку, яка подана голкою і човниковою нитку (об'єкт суміжного базового класу) на шпульці човника для утворення вузлика переплетення двох ниток.

3. *Object=ниткопритягувач* або **AttractThread=object** – подати нитку голці для утворення *і-го* стібка, потім човнику, зменшити розмір обведеної навколо шпулько держателя голкової нитки до розміру вузлика, затягнути утворений вузлик в середину пакету зшиваємих матеріалів (об'єкт суміжного базового класу) і змотати з котушки голкову нитку довжиною, яка дорівняє довжині нитки зароблений в *і-ий* стібок

4. *Object=зубчаста рейка* або **BottomFoot=object** – слідкує за положенням вістря голки (об’єкт цього ж базового класу) відносно верхньої поверхні пакету текстильних матеріалів, які й знаходиться між притискною лапкою (об’єкт цього базового класу) і голковою пластиною (об’єкт цього базового класу) та перемістити зшиваєми матеріали на задану довжину стібка із ниток, коли вістря голки вийде з матеріалу (об’єкту суміжного базового класу).

5. *Object=притискна лапка* або **PresserFoot=object** – при активному робочому процесі постійно взаємодіє тільки з об’єктом суміжного класу Cloth=object.

6. *Object=голкова пластина* або **PlateNeedle=object** – виконує роль посередника для реалізації робочого процесу і також постійно взаємодіє тільки з об’єктом суміжного класу **Cloth=object** і періодично з BottomFoot=object свого класу.

7.*Object=регулятор натягу голкової нитки* або **SensorThread=object** – виконує функції датчика натягування ниток, який знаходиться в пасивному стані (увімкнений) на протязі всього робочого процесу. Він переходить автоматично в активний стан, а саме не затискує голкову нитку спожиту ниткопритягувачем при утворенні наступного стібка до моменту отримання команди від вузлика із ниток і матеріалу (об’єктів суміжного базового класу) при куті повороту головного валу машини, коли цей вузлик припиняє переміщення по вертикалі і зупинитися в середині пакету деталей з текстилю.

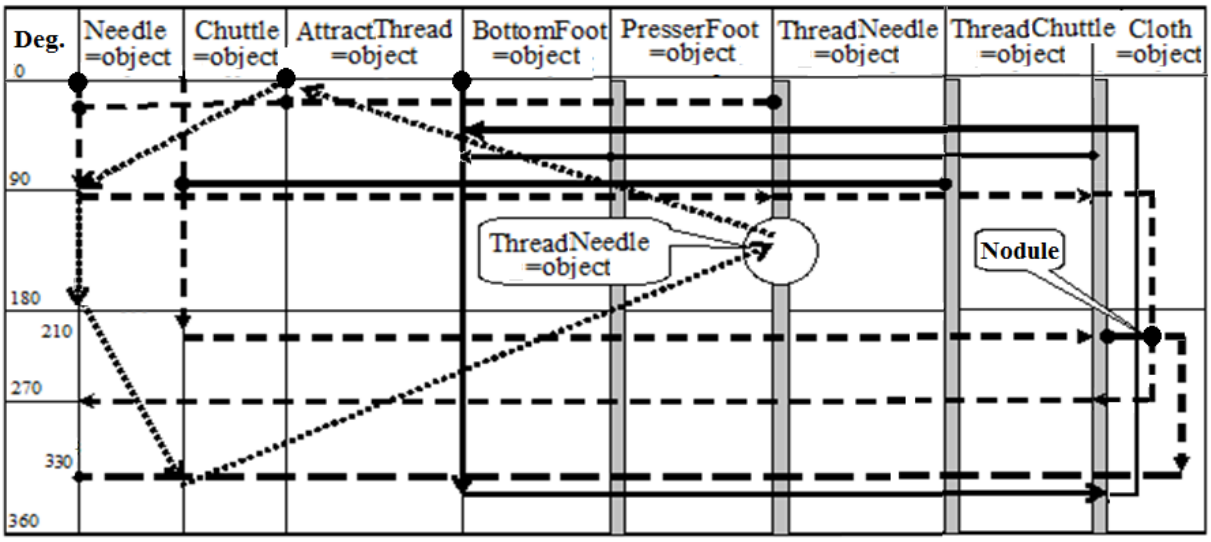


Рис. 3.7. Об’єктно-орієнтована діаграма взаємодії об′єктів технологічної машини на прикладі човникової швейної машини загального призначення з жорсткою системою керування типу «розподільний вал»

Формалізованими об’єктно-орієнтованими обов’язками делегованими механіко-технологічною системою для базового класу Object=сировина (рис.3.2) або **RawMaterial=object**; {об’єктний тип} є наступні:

1. *Object=нитка* породжує *Object=голкова нитка* або **ThreadNeedle=object** та *Object=човникова нитка* або **ThreadChuttle=object**.

ThreadNeedle=object – з позиції об′єктно-орієнтованого аналізу цей об’єкт, як і інші об’єкти цього класу є передаточними ланками між об’єктами суміжного базового класу Tool=object. Голкова нитка є пружною ланкою, яка вказівкою-командою (1) передає для Needle=object команду провести петлю скрізь матеріал, а для Chuttle=object вказівкою-командою (2) передає сигнал по положенню головного валу о своєї взаємодії з носиком човника при переміщенні голки із крайнього нижнього положення на величину S_0 = const.

Ця вказівка-команда (2) передається за допомогою матеріалу, в якому знаходиться голка за рахунок різності сили тертя між матеріалом і тієї ділянки нитки, яка прихована в довгому жолобку голки і сили тертя ділянки нитки з протилежного боку голки цієї ж нитки, яка безпосередньо знаходиться в матеріалі.

Вказівку-команду (3) голкова нитка передає для **AttractThread=object** про необхідність скорочення довжині розширеної петлі-напуска і втягуванні вузлика переплетення із двох ниток в матеріал.

Вказівка-команда (4) формується для **SensorThread=object** про необхідність подачі нитки для наступного машинного стібка.

Для **ThreadChuttle=object** системою делеговані обов’язки човникової нитки, які інкапсульовані в обов’язках **ThreadNeedle=object** після утворення вузлика із двох ниток.

2. *Object=тканина* або **Cloth=object** – є пружнім передаточною ланкою між **PresserFoot=object** і **PlateNeedle=object**, тобто є

передаточною ланкою двома об’єктами суміжного базового класу. Її обов’язки повинні бути сформульовані для системи у списку відповідних наступних вказівок (команд) голці, ниткопритягувачу, зубчастій рейки і регулятору натягування голковій нитки:

Вказівка-команда (1) голки – проведи скрізь мене голкову нитку у вигляді симетричної петлі і коли обидві гілки петлі досягнуть максимальній довжині, тоді з однієї сторони нитки потрібно призупинись для утворення петлі-напуска.

Вказівка-команда (2) для ниткопритягувача інкапсульована в **AttractThread=object**.

Вказівка-команда (3) зубчатої рейки – переміщуй **Cloth=object** на задану довжину стібка, так як голка вже вийшла із матеріалу і його переміщенню вперед ніщо не перешкоджає.

Вказівка-команда (4) для **SensorThread=object** – вузлик із двох ниток вже втягнутий усередину **Cloth=object** і якщо зараз не буде ослаблена натягування **ThreadNeedle=object**, тоді не із чого буде отримувати наступний стібок або голкова нитка обірветься, чому присвоєно пасивний стан робочого процесу.

Формалізовані об’єктно-орієнтованими обов’язки, які делеговані механіко-технологічною системою базовому класу Object=механізм або **Mechanism=object** (рис. 3.2) для Object=ланка або **Link=object**; {об′єктний тип} технологічної машини, розглянемо на прикладі однієї з типових ланок різних механізмів цільового призначення. В якості такої ланки вибираємо шатун **Connecting=object** і з позиції «шатунності» розглянемо інші типові ведучі передаточні і ведені ланки важільних механізмів, як об’єкти що інкапсульовані в **Link=object**. Зміст «шатунності» полягає в тому, що будь-яка ланка може бути задана на площині двома точками, відрізками прямої заданої довжини L = const і кутом φ_0 в прийнятій системі координат. Для кривошипу (**Crank=object**) і ексцентрика (**Eccentrik=object**) потрібно додатково вказати, що одна точка з заданими координатами є стійкою (**Bearing=object**) і зміна кута $\varphi_i \geq 360^0$, для кулачків (**Wobbler=object**) - теж саме, тільки $L \neq$ const. Для коромисла (**Rocker=object**), як для кривошипу тільки $\varphi_i < 360^0$.

При цьому механіко-технологічна система з позиції алгоритму обєктно-орієнтованого проектування делегує об’єктам **Crank=object**,

Crank=object (кривошип), **Eccentrik=object (**ексцентрик) і **Cam=object** (кулачок) виконання наступних обов'язків.

Кривошип, ексцентрик, кулачок, коромисло в геометричної моделі мають задані геометричні розміри (*l = const*), координати стійкі (*x0* та *y0*) і початковий кут φ_0 положення. Для ведучої ланки кут повороту задається в прирощеннях для визначення швидкості і прискорення в відкладеному рішенні. Такі ланки повинні генерувати повідомлення на кожному виконаному циклі (обороті головного валу) робочого процесу. При цьому робота цільових механізмів голки, ниткопритягувача, зубчатої рейки ідентифікується повідомленнями про нові координати x_i і y_i за допомогою зміни координат осі пальця кривошипу або осі ролика товкача кулачкового механізму, які утворюють обертальну кінематичну пару 5 класу з першою головкою шатуна, або з поверхнею кулачка в залежності від узагальненої координати φ_i – кута повороту ведучої ланки. Таким чином, обов'язки будь-якого із перелічених ланок є генерування двох гармонічних функцій, в проекціях на осі координат і зміщених по фазі на π/2 : $x_i = L\cos\varphi_i$ і $y_i = L\sin\varphi_i$.

Кривошип, ексцентрик, кулачок, коромисло в геометричної моделі мають задані геометричні розміри (***l = const***), координати стійкі (***x0*** та ***y0***) і

Перша голівка шатуна сприймає ці координати без змін, якщо не враховується знос ведучої ланки або його посередника (наприклад, пальця кривошипу) і які з урахуванням зносу можуть бути розглянути ні як відкладені рішення на першій стадії проектування. Другій голівці шатуна (друга крайня точка відрізку прямої постійної довжини) в 4-х-ланкових типових важільних механізмах системою делегуються функція положення першої голівки шатуна з врахуванням кінематичної геометричної константи $\lambda = \frac{r}{l}$. Цієї точки шатуна або центру мас делегується обов'язки першої головки шатуна. В 5-ти ланкових важільних механізмів з двома ведучими ланками другої точки першого шатуна і другої точки другого шатуна делегуються функції положення двох ведучих ланок. Шатун може мати додаткові відростки (поводки), утворюючи або 3х-поводкову групу Ассура, або утворюючі, наприклад, вічко ниткопритягувача, яке переміщується по траєкторії - шатунній кривій.

На основі об'єктно-орієнтованого проектуванню механізмів, починаючи з другої голівки шатуна, шатун делегує свої функції положення

в плоско-паралельному русі всім кінематичних ланцюгів, в які входить цей шатун і які визначають для цього шатуна першу і другу передаточні функції. Тому швидкості і прискорення таких шатунів визначаються синхронним і одночасним описом ***i*** та ***i+n*** положень всіх інших ведених ланок механізму за допомогою ***object=метод*** в програмному коді. Такий опис виконується на основі аналітичних залежностей, аргументи яких інкапсульовані в цей же об′єктно-орієнтований метод розрахунку.

Команди , як реакції на основні обов'язки робочих інструментів і ланок цільових механізмів технологічних машин будуть реалізовані, якщо і робочий процес для цих механізмів буде також спроектований на основі об′єктно-орієнтованого аналізу і об′єктно-орієнтованого синтезу.

За аналогією обов'язками повзуна (***Slider=object***), каменю куліси (***Slot=object***), поршню циліндру, голководу і інших ланок, які утворюють поступальну кінематичну пару 5го класу, з назвою важіль (***Arm=object***) або поводок (***Tang=object***) в класі об'єктів ***Tool=object*** - є повідомлення координат двох точок, з'єднаних відрізком прямої (направляючої), яка може бути нерухомою або відхилятися відносно однієї із точок направляючої.

3.6. **Приклад об'єктно-орієнтованого аналізу і синтезу мехатронних циклових систем**

Після декомпозиції циклової механіко-технологічної системи на елементи можна сформулювати наступні *об'єктно-орієнтовані обов'язки* елементів і функції делегування (передачі) відпрацьованих обов'язків наступному суміжному елементу циклової системи.

Розглянемо мехатронну систему поштучного відокремлення гнучких деталей низу взуття зі стосу для автоматизації взуттєвих машин. Робочий процес реалізується БІстабільною системою керування двома пневмоциліндрами двосторонній дії, які розташовані в ортогональних площинах.

Зменшення сил зчеплення деталей в стосі при поштучному їх відокремлення зі стопи знизу у магазинному завантажувальному пристрою (МЗП) може бути досягнута за рахунок імпульсу сили (удару) по нижньої деталі.

У відповідності до функціонального графу « $1 \to 3 \to \overline{1} \to 2 \to \overline{3} \to \overline{2}$ » (рис.3.8) після включення/виключення пневмоциліндру 1 (рис.3.9) імпульс

сили діє на стос деталей знизу, одночасно програмно включається пневмоциліндр 2, якій просуває нижню деталь у валки 3 і 4 транспортного модуля технологічної машини при виконанні умови метричного синтезу:

$$h_2 \geq L_1, \tag{3.1}$$

де h_2 – величина ходу штоку пневмоциліндру 2;

L_1 – відстань між торцем нижньої деталі в стосі і валиками 3 і 4.

.

Із програмною затримкою часу 1 секунда пневмоциліндр 2 вимикається і цикл повторюється. Наявність деталей в МЗП контролюється кінцевим вимикачем – нормально відкритим контактом кнопки імітації S_{im}.

Функціональний граф «$1 \to 3 \to \bar{1} \to 2 \to \bar{3} \to \bar{2}$» і таблиця логічних сигналів відповідності наведені на рис.3.8, а комбінована схема автоматизованого завантажувального пристрою наведена на рис. 3.9, де прийняті наступні умовні позначення обраних елементів механічної і енергетичної складових циклової системи керування:

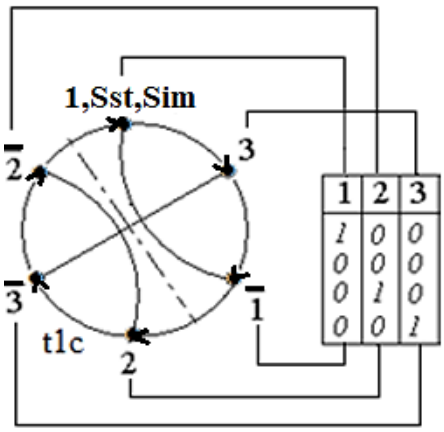


Рис. 3.8. Функціональний граф і таблиця відповідності:
1 і 2 – виконавчі механізми; 3 – програмний елемент пам'яті

На рис. 3.9 прийняти наступні позначення елементів:

1 – пневмоциліндр з БІстабільним керування ($Y1, Y\bar{1}$) для реалізації ударного імпульсу сили;

2 – пневмоциліндр з БІстабільним керування ($Y2, Y\bar{2}$) для реалізації виведення нижньої деталі стосу і подавання її у валики 3 і 4;

3 – притискний валик;

4 – транспортуючий валик;

5 – магазинний завантажувальний пристрій;

6 – деталі низу взуття;

7 – перший БІстабільний пневморозподільник (тип 5/2) з електромагнітним керуванням;

8 – кінцевий вимикач $X_{\bar{1}}$ контролю початкового положення штоку пневмоциліндру 1 (ПЦ1);

9 – кінцевий вимикач X_1 контролю кінцевого положення штоку пневмоциліндру ПЦ1;

10 – другий БІстабільний пневморозподільник (тип 5/2) з електромагнітним керуванням;

11 – кінцевий вимикач $X_{\bar{2}}$ контролю початкового положення штоку пневмоциліндру 2 (ПЦ2) ;

12 – кінцевий вимикач X_2 контролю кінцевого положення штоку пневмоциліндру ПЦ2;

13 – дросель для регулювання швидкості ударного імпульсу.

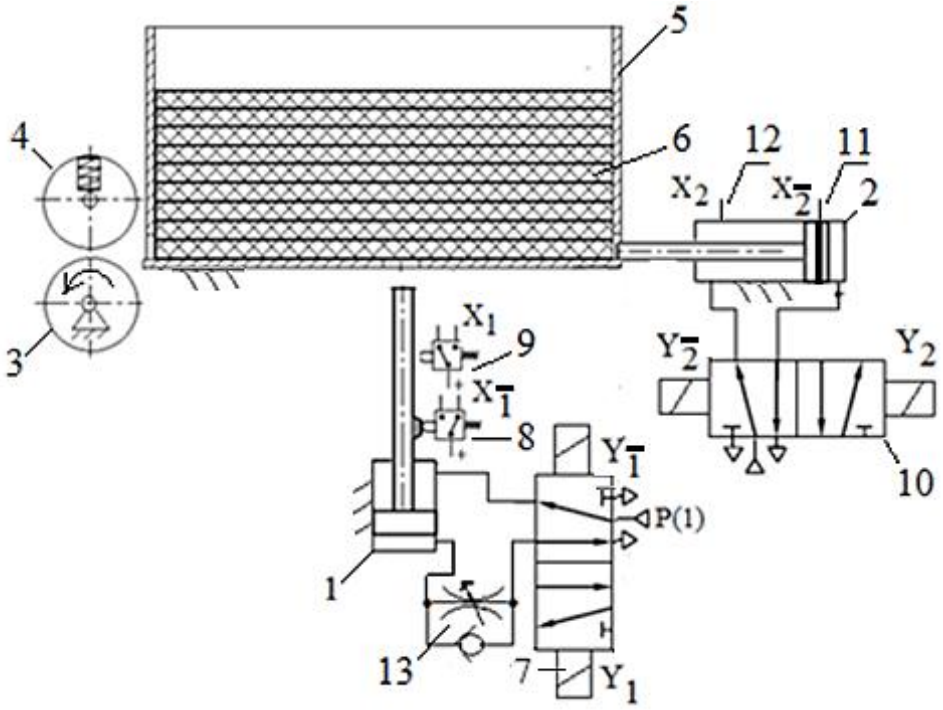
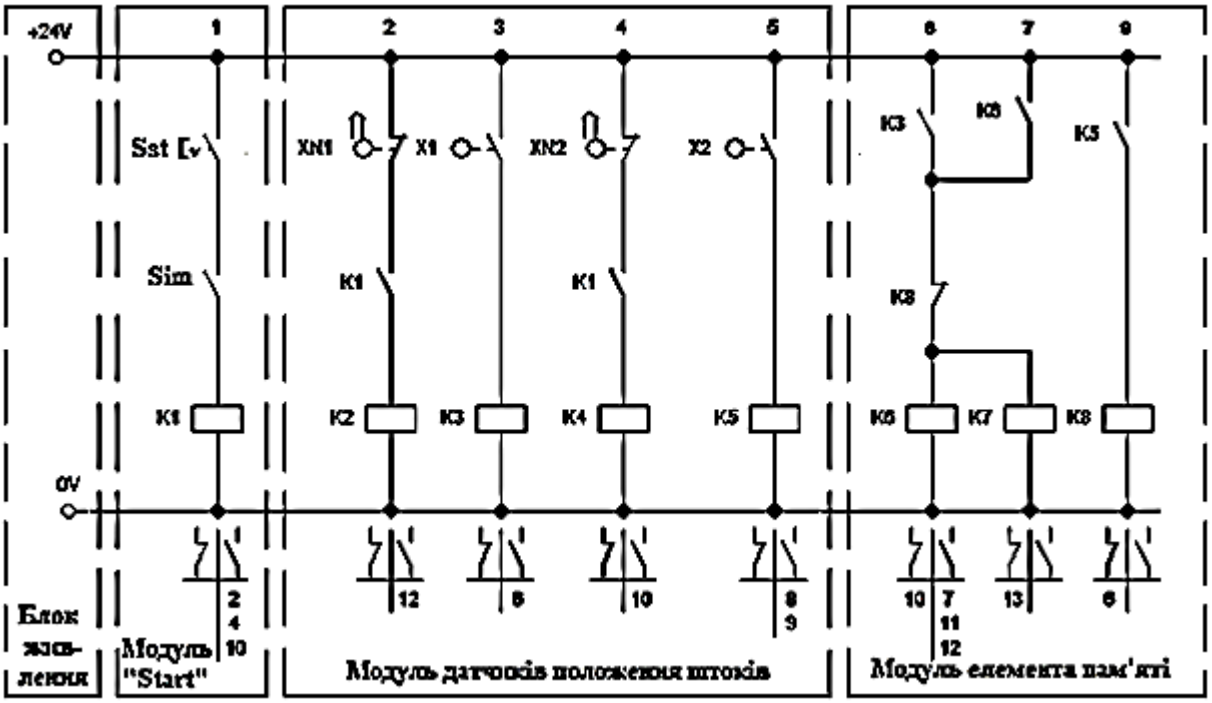
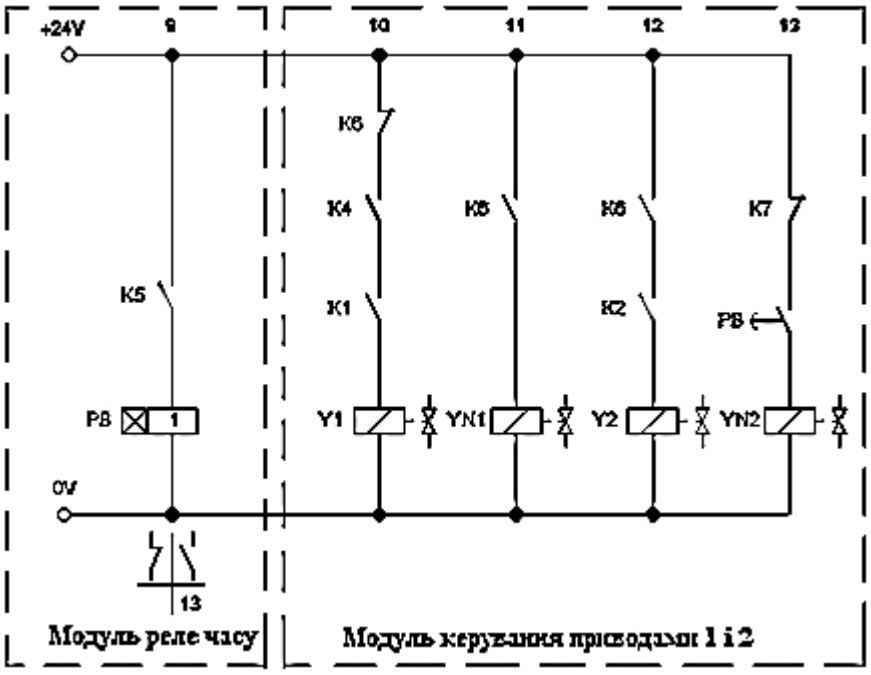


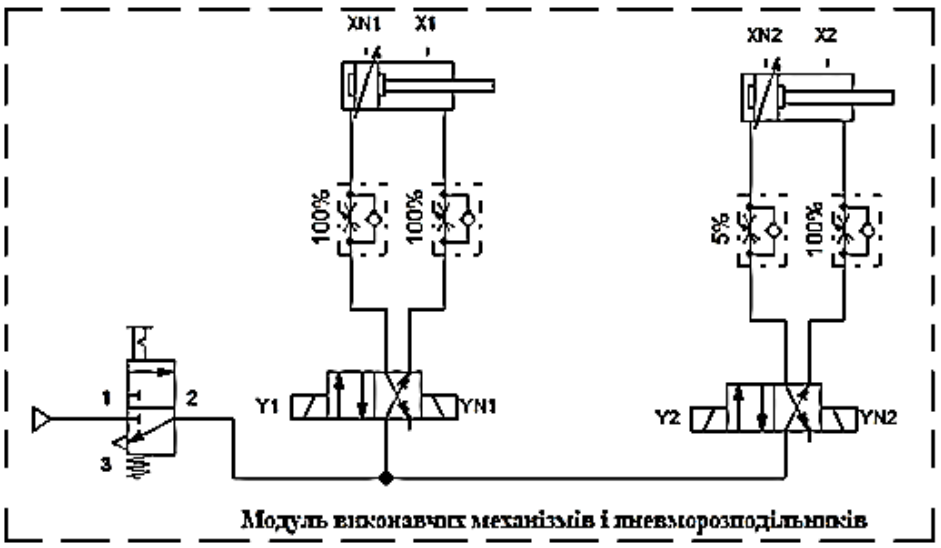
Рис. 3.9. Комбінована принципова схема автоматизованого завантажувального пристрою


+24V
1
2
3
4
5
6
7
8
Sst
Sim
XN1
X1
XN2
X2
K1
K2
K3
K4
K5
K6
K7
K8
0V
Блок живлення
Модуль "Start"
Модуль датчиків положення штоків
Модуль елемента пам'яті

а


+24V
9
10
11
12
13
K1
K2
K4
K5
K6
K7
PB
Y1
YN1
Y2
YN2
0V
Модуль реле часу
Модуль керування приводами 1 і 2

б



в

Рис. 3.10. Електромеханічні (а, б) і пневматичні (в) модулі схем БІстабільного керування магазинним завантажувальних пристроєм взуттєвих машин

Для програмування системи БІстабільного керування автоматизованого завантажувального пристрою на рис.3.9 в таблиці 3.1 наведені зведені характеристика і символьні імена обраних елементів модулів мехатронної системи.

Таблиця 3.1 Символьні імена елементів модулів мехатронної системи

Ударний імпульс сили по деталям стосу знизу			Виведення нижньої деталі стосу			Елементи		
Пневмо-циліндр 1	Елемент керування	Конт-роль штоку	Пневм оцилін др 2	Елемент керування	Конт-роль штоку	Пам' яті	Затр имки часу	Кнопки
Схема комбінована (електропневматіки)								
Началь не по-ложенн я – шток втягну-тий.	Пневмокл апан 5/2 з БІ-стабільни м електрома г-нітним керуванн ям Y1 та YN1	Кінцев і вимик ачі (XN1) та (X1)	Началь не по-ложенн я – шток втягну-тий.	Пневмокл апан 5/2 з БІ-стабільни м електрома гнітним керуванн ям Y2 та YN2	Кінцев і вимик ачі (XN2) та (X2)	Дода ткове реле з пріор итето м на вими канн я (3)	Реле часу (T1)	Sst вмикан -ня систем и; Sim імітація наявно сті деталей в МЗУ

У початковому положенні поршні і штоки пневмоциліндрів 1 і 2 втягнути. Послідовність дій « $1 \to 3 \to \overline{1} \to 2 \to \overline{3} \to \overline{2}$ » в циклі функціонального графу утворена після додавання елементу пам'яті 3 (рис.3.8), якій додатково введений для усунення інформаційної невизначеності у зв'язку з існуванням у початковому графі двох підграфів розділених лінією невизначеності (штрих пунктирна лінія на рис. 3.8).

Складаємо наступні логічні вирази причино-наслідкових зв'язків для команд керування, які супроводжуються відповідним фрагментом програми для контролера і коментарем до нього:

Програма БІстабільного керування автоматизованим завантажувальних пристроєм для варіанту з використання контролера:

$$Y_1 \Leftarrow X_{\overline{2}} \cdot X_{\overline{3}} \cdot Xst \cdot Xim; \qquad (3.2)$$

```
IF     XN2          ' якщо виконується умова(2)
   AND XN3
   AND Sst
   AND Sim)
THEN    SET   Y1   ' тоді включити пряму команду для
ПЦ1

RESET YN1       ' вимкнути зворотну команду для ПЦ1
```

$$Y_{\overline{1}} \Leftarrow X_3; \qquad (3.3)$$

```
IF  X3             ' якщо виконується умова (3)
THEN SET YN1       ' тоді включити зворотну команду для
ПЦ1
   RESET  Y1       ' вимкнути пряму команду для ПЦ1
```

$$Y_2 \Leftarrow X_{\overline{1}} \cdot X_3 ; \qquad (3.4)$$

```
IF XN1 AND X3      ' якщо виконується умова (4)
THEN SET   Y2      ' тоді включити пряму команду для ПЦ
2
    RESET YN2      ' вимкнути зворотну команду для ПЦ2
```

$$Y_{\overline{2}} \Leftarrow X_{\overline{3}}; \qquad (3.5)$$

```
IF  XN3            ' якщо виконується умова (5)
THEN SET  YN2      ' тоді включити зворотну команду для
ПЦ2
```

```
     RESET Y2     ' вимкнути пряму команду для  ПЦ2
```

$$Y_3 \Leftarrow X_1; \qquad (3.6)$$

```
IF   X1             ' якщо виконується умова (6)
THEN SET X3         ' тоді включити прапор включеного
                  ' стану елемента  пам'яті 3 (рис.3.8)
RESET XN3           ' вимкнути прапор виключеного
стану
                  ' елемента пам'яті затримки часу 1 с
RESET XT1           ' вимкнути прапор покриття таймера
T1

IF N XT1 AND X2     ' якщо прапор покриття таймера
                    ' активний та контакт кінцевого
                    ' вимикача X2 замкнений (X2=1)
THEN SET T1         ' тоді включити таймер T1
    WITH 1s         ' із затримкою часу 1 с
SET XT1             ' включити прапор покриття таймера
T1
RESET X3            ' вимкнути прапор  включеного
стану
                    ' елемента  пам'яті 3
```

$$Y_{\bar{3}} \Leftarrow X_2 \; ; \qquad (3.7)$$

```
IF NT1 AND XT1      ' якщо таймер закінчив роботу
(T1=0)
                    ' і прапор покриття таймера
(XT1=0)
THEN SET XN3        ' тоді включити прапор вимкнутого
                    ' стану елемента пам'яті  3
```

Для програмної реалізації команд (3.2)…(3.7) в таблиці 4.2 наведені абсолютні та символьні імена операндів системи циклового програмного. На рис. 3.11 наведена електрична схема Бістабільного керування автоматизованим завантажувальних пристроєм взуттєвих машин з використання контролера.

Таблиця 4.2 AllocationList

Тип	Абсолютне ім'я (***Abs***)	Символьне ім'я (***Simb***)	Коментар
Output	**O0.0**	**Y1**	Котушка соленоїда Y1 під струмом (YN1=0)
Output	**O0.1**	**YN1**	Котушка соленоїда YN1 під струмом (Y1=0)
Output	**O0.2**	**Y2**	Котушка соленоїда Y2 під струмом (YN2=0)
Output	**O0.3**	**YN2**	Котушка соленоїда YN2 під струмом (Y2=0)
Input	**I0.0**	**XN1**	Шток приводу 1 внизу
Input	**I0.1**	**X1**	Шток приводу 1 вверху (удар)
Input	**I0.2**	**XN2**	Шток приводу 2 втягнутий
Input	**I0.3**	**X2**	Шток приводу 2 висунутий (зсув нижньої деталі від стосу)
Input	**I0.4**	**Sst**	Сигнал вмикання системи (кнопка Start)
Input	**I0.5**	**Sim**	Сигнал імітації наявності деталей в МЗУ(нормально відкритий контакт Sim)
Flag	**F0.0**	**XN3**	Прапор вимкнутого елемента пам’яті 3
Flag	**F0.1**	**X3**	Прапор включеного елемента пам’яті 3
Flag	**F0.2**	**XT1**	Прапор покриття таймера(запобігає повторному вмиканню)
Flag	**FX1**	**T1**	Таймер

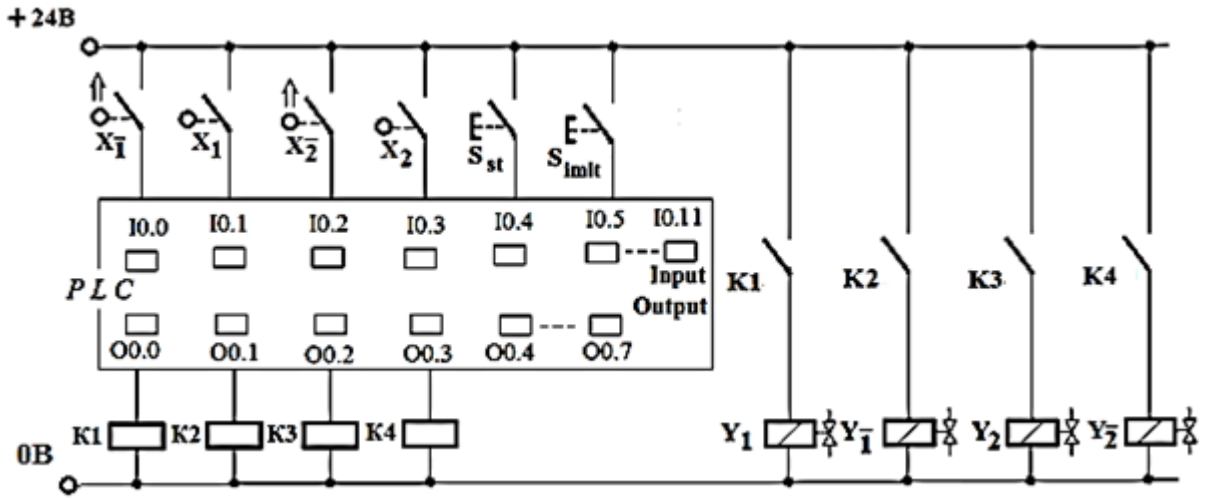


Рис. 3.11. Електрична схема Бістабільного керування автоматизованим завантажувальних пристроєм взуттєвих машин з використанням контролера

Після завантаження МЗП деталями низу взуття (**Xim** =**1**) і натисканні кнопки старту (**Xst = 1**) при замкненому контакті $\mathbf{X_{\bar{2}} = 1}$ і вимкненому елементі пам'яті $\mathbf{X_{\bar{3}} = 1}$ починається наступний цикл роботи АЗП:

виконується пряма команда $Y_1 := Y1 = 1$ при який шток пневмоциліндру 1 (ПЦ1) швидко висувається і відбувається удар штоком по стосу деталей знизу. При цьому виникає сигнал контролю $X_1 := X1 = 1$ другого крайнього положення штоку;

вмикається елемент пам'яті $X_3 := X3 = 1$ і виконується зворотна команда $Y_{\bar{1}} := YN1 = 1$. Шток пневмоциліндру 1 повертається у початкове положення і сигнал контролю першого крайнього положення штоку $X_{\bar{1}} := XN1 = 1$;

потім виконується пряма команда $Y_2 := Y2 = 1$ і шток пневмоциліндра 2 (ПЦ2) висувається з першого крайнього положення. Відбувається подавання нижньої деталі зі стосу у валки 3 і 4 АЗП і сигнал контролю другого крайнього положення штоку $X_2 := X2 = 1$. Вмикається таймер (Т1) із затримкою часу на 1 секунду;

вимикається елемент пам'яті $X_{\bar{3}} = 1$ і відключається таймер;

виконується зворотна команда $Y_{\bar{2}} := YN2 = 1$ і шток пневмоциліндру 2 повертається у початкове положення, а сигнал контролю першого крайнього положення штоку $X_{\bar{2}} := XN2 = 1$.

Якщо кнопка фіксацією була натиснута (**Xst = 1**) і в МЗП є детал, то цикл повторюється і подається наступна деталь низу взуття. Якщо **Xim** = 0, тоді наступний цикл не починається і це є сигналом на поповнення МЗУ деталями. Якщо повторно натиснути на кнопку з фіксацією (**Xst = 0**), тоді система допрацює цикл і зупиняється.

Контрольні питання до розділу 3.

1.Чим відрізняється об'єктно-орієнтований аналіз від об'єктно-орієнтованого синтезу ?

2. Які об'єкти базового класу робочих інструментів ***Tool=object*** підлягають розгляду при об'єктно-орієнтовані аналізі крає обметувальної машини ?

3. Які об'єкти базового класу робочих інструментів ***Tool=object*** підлягають розгляду при об'єктно-орієнтовані аналізі плоско в'язальної машини типу ПВК ?

4. Які об'єкти базового класу сировина ***RawMaterial=object*** підлягають розгляду при об'єктно-орієнтовані аналізі машин човникового стібка ?

5. Які об'єкти базового класу сировина ***RawMaterial=object*** підлягають розгляду при об'єктно-орієнтовані аналізі основов'язальної машини ?

6. Які об'єкти базового класу сировина ***Link=object*** підлягають розгляду при об'єктно-орієнтовані аналізі кривошипно-коромислового механізму ниткоприягувача ?

7. Якій зміст мають команди **`SET YN2`** і команд **`RESET Y1`** ?

4. ПРОГРАМОВАНІ ВИКОНАВЧІ МЕХАНІЗМИ ОБЕРТОВОЇ ДІЇ МЕХАТРОННИХ І РОБОТОТЕХНІЧНИХ СИСТЕМ

4.1. **Структура виконавчих механізмів мехатроніки на засадах електроприводу**

Машина відрізняється від механізму тим, що машина, як правило, має електропривод. Якщо механізм має привод, тоді такий механізм має назву «*механізм з індивідуальним приводом*», а в мехатронних системах та в автоматизованих технологічних машинах має назву «*програмно-керований механізм*».

В неавтоматизованих технологічних машинах легкої промисловості з жорсткими системами керування застосовується електропривод на засадах асинхронного електродвигуна з редуктором (в'язальні машини) і асинхронного електродвигуна з муфтою зчеплення ***Clutch Motor*** (двигун зчеплення). Відомо, що в асинхронному електродвигуні котушки статора (пари полюсів) зміщені по фазі на кут 120^0 і тому створюють обертовий магнітний потік частотою *n*:

$$n = \frac{60 \cdot f}{p} \quad [\text{об/хв}],$$

де $f = 50$ Гц – частота живлення змінним струмом;
p – кількість пар полюсів статора.

Частота n з урахуванням ковзання визначає частоту обертання ротора асинхронного електродвигуна. Змінювати частоту обертання ротора можна, якщо змінювати в чисельнику f або в знаменнику p . В першому випадку для плавного регулювання частоти обертання ротора потрібне додаткове застосування технічних засобів регулювання частоти, а у другому випадку для ступінчастого регулювання потрібно збільшені затрати дроту з міді на виготовлення додаткових котушок статора і засобів їх переключення при пуске і при гальмуванні асинхронного електродвигуна з короткозамкненим ротором.

В автоматизованих технологічних машинах і верстатах машинобудування з гнучкими системами керування використовують

виконавчі механізми зворотно-поступової дії або виконавчі механізми обертової дії (*електродвигуни*).

До програмно керованих виконавчих механізмів зворотно-поступової дії, які застосовуються в технологічних машинах галузі і верстатах машинобудування відносяться:

- пневмоциліндри і гідроциліндри;
- лінійні електродвигуни;
- виконавчі механізми обертового руху з передаточним механізмом та інші.

В системах мехатроніки виконавчі механізми зворотно-поступової дії або обертової дії підключаються до портів «***Output***» (портів виходів) програмуемого логічного контролера CNC-верстату або комп'ютерної технологічної машини.

Автоматизація технологічних машин і апаратів з прямим електроприводом (*англ.* ***Direct Drive Motor***) передбачає розробку гнучкої системи керування для автоматизації електроприводу без застосування передаточних важільних механізмів . Якщо в технологічних машинах використані важільні механізми з індивідуальним виконавчим механізмом обертової дії на засадах крокового/вентильного двигуна (***Stepper Motor***) чи серводвигуну (***Servo Motor***) або використані виконавчі механізми зворотно-поступової дії, тоді такі важільні механізми переходять в клас «***програмно-керованих механізмів***». Сучасна назва таких механізмів ***«механізми з інтелектом»*** і вони є основою робототехніки.

Розробка (*проектуванні та виготовлення*) технологічних машин легкої промисловості з *програмно-керованими механізмами* дозволяє багатоланковий важільний механізм цільового призначення замінити на 4-ланковий механізм з індивідуальним автоматизованим приводом. При цьому функція положення веденої ланки з одної або декілька зупинками за один оберт головного валу (один цикл) досягається засобами програмування контролера електроприводу. Доцільність такої заміна відноситься як до плоских, так і до просторових важільних механізмів, а також до більшості одно крокових і багатокрокових кулачкових механізмів технологічних машин легкої промисловості.

Електропривод автоматизованих машин включає сукупність наступних трьох складових:

1 – електродвигуна;

2 – засобів передачі обертового руху від електродвигуна до головного валу машини. Це редуктори, муфти, пасові та інші гнучкі і ланцюгові передачі. Якщо електродвигун напряму з'єднаний з головним валом машини без додаткових засобів передачі і тоді такий електропривод має назву «***прямий привод***» і технологічна машина має назву, наприклад «*швейна машина з прямим приводом*», «*пральна машина з прямим приводом*» тощо;

3 – засобів комутації . Це кнопки пуск/стоп, кінцеві вимикачі, датчики кута повороту для зворотного зв'язку, технічні засоби для захисту електродвигуна від перевантажень, коротких замкнень, перегріву.

До поширених виконавчих механізмів обертової дії, які сполучаються з контролером є наступні електроприводи автоматизованих машин легкої промисловості:

- *привод з асинхронним електродвигуном і частотним регулюванням* (складається з асинхронного електродвигуна з короткозамкненим ротором і перетворювача частоти);
- *привод з колекторними електродвигунами постійного струму* (має перетворювач постійного струму);
- *кроковий електропривод* (має драйвер цифрового керування без зворотного зв'язку);
- *сервопривод* (має драйвер керування і датчик для зворотного зв'язку).

За аналогією виконавчі механізми зворотно-поступової дії мехатронних систем також мають три складових:

1 – пневмоциліндр (гідроциліндр);

2 – засоби передачі (пневморозподільник/гідророзподільник);

3 – засоби комутації (кінцеві вимикачі, сенсори і інші засоби сигналізації стану включено/вимикнемо).

Приводи з асинхронним електродвигуном з частотним регулюванням – це система з двох компонентів прямої дії (без зворотного зв'язку), а саме асинхронного електродвигуна з короткозамкненим ротором (надійність і мала вартість зробили такий електродвигун самим розповсюдженим електродвигуном неавтоматизованих технологічних машин) і блока керування частотою 3-фазного живлення (рис. 4.1,а).

У розімкненої системі керування без зворотного зв’язку між електродвигуном (або навантаженням) і блоком керування у випадках зміни одного з параметрів (наприклад електричної напругі, струму, положення ротора) робота системи змінюється.

У замкненої системі керування (рис. 4.1,б) при зміні параметрів виконується їх автоматична корекція (компенсація) по каналу зворотного зв’язку і запрограмована контролером функція роботи не змінюється.

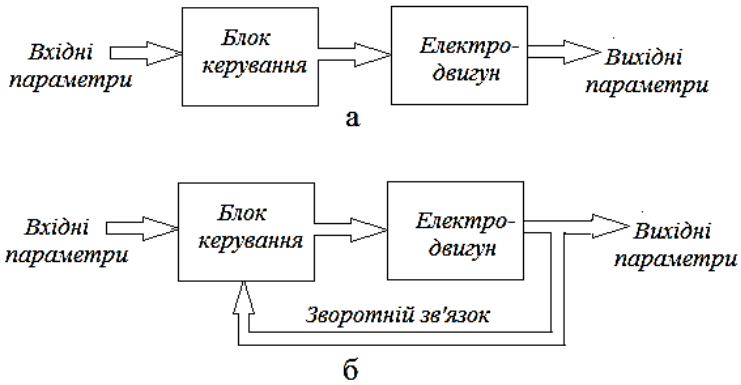


Рис. 4.1. Структурні схеми виконавчих механізмів мехатроніки: а – прямої дії (розімкнена система керування); б – система керування зі зворотним зв'язком (замкнена система керування)

Приводи з електродвигунами постійного струму (поширена назва «***колекторні двигуни***») використовується в машинах і апаратах, в яких потрібно забезпечити регулювання швидкості у широкому діапазоні і коли зміна величини навантаження незначно впливає на частоту обертання.

Приводи з кроковими електродвигунами широко використовується в CNC-машинах (машинах з **ЧПК** – **Ч**исловим **П**рограмним **К**еруванням), комп’ютерних плоско в’язальних машинах, розкрійних агрегатах з ЧПК і в металообробних верстатах з ЧПК. Кроковий двигун це цифровий безколекторний електродвигун постійного струму, який перетворює електричні імпульси в механічний рух ротора кроками. Кожний імпульс відповідаю певному куту повороту (одному кроку) валу ротора електродвигуна. Крокові двигуни зі зворотним зв’язком від датчика Холла для кута повороту ротора мають назву *вентильні двигуни*.

Сервоприводи є програмуємими приводами, які побудовані на засадах безколекторних двигунів постійного стуму (рис.4.2) або на засадах синхронних двигунів змінного струму. Серводвигуни мають структуру по

схемі на рис. 4.1,б , де зворотний зв'язок виконує датчик зворотного зв'язку у вигляді, відповідного *енкодера* або *резольвера.* Резольвер це обертовий трансформатор, який має одну рухому котушка закріплену на валу ротора синхронного електродвигуна, а дві нерухомих котушки (обмотки) зсунути по фазі на кут π/2.

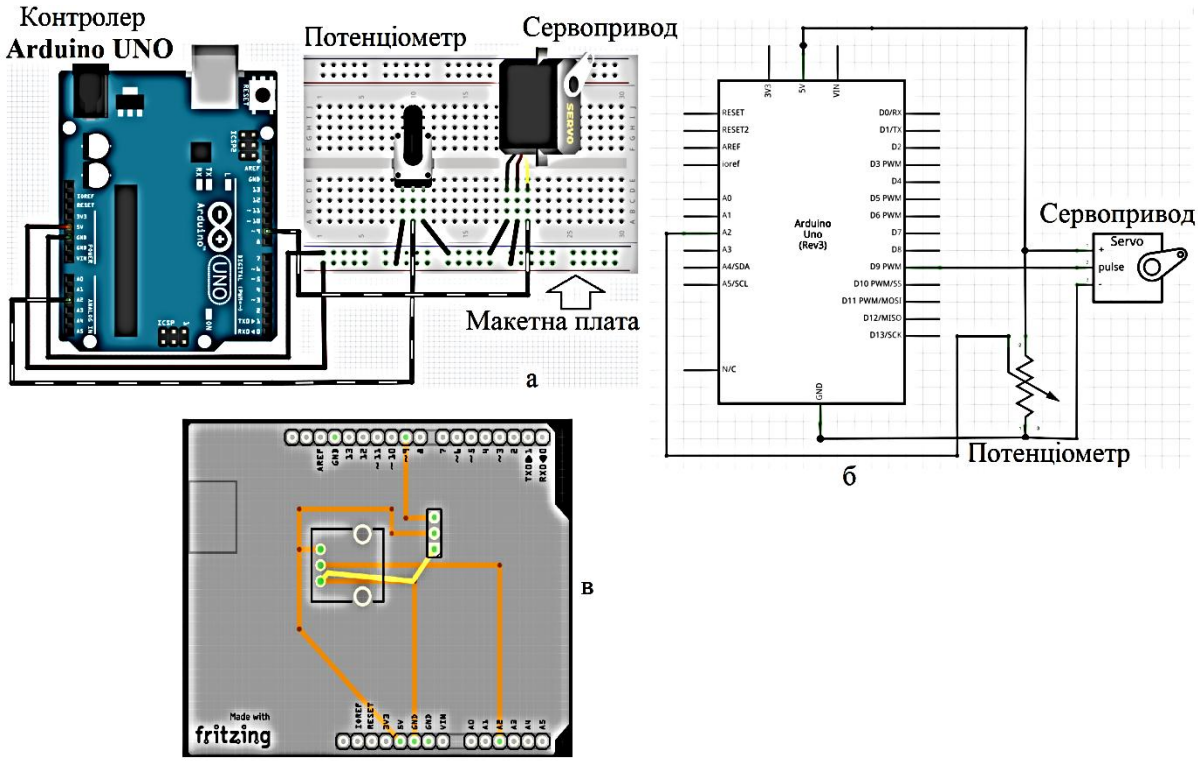


Рис. 4.2. Приклади монтажної схеми (а), електричної принципової схеми (б) і друкованої схеми (в), побудованих в програмному середовищі **Fritzing** для керування сервоприводом **SG-90** з використанням контролера **Arduino UNO** та потенціометра

Сервопривод на засадах крокового двигуна є також поширеним приводом автоматизованих машин легкої промисловості і CNC-верстатів машинобудування, тому що він програмно забезпечує перетворення любої функції положення механізму. Наприклад, може виконувати функції жорсткого багатокрокового кулачка циклових швейних машин-напівавтоматів. Тоді такий багатокроковий жорсткий програмоносій стає гнучким програмоносієм і буде мати назву «***електронний кулачок***». Сервопривод може реалізувати різні функції зміни швидкості приводного моменту по наперед заданому закону або повертати ведену ланку на потрібний кут за стрілкою годинника або проти.

Комп'ютерно-інтегровані швейні машини і машини-автомати використовують автоматизований електропривод, що має різну архітектуру (модифікації М1, М2, М3, М4) і які побудовані на мікропроцесорних системах керування мехатроніки :

Модифікація ***М1*** – «Автоматизований електропривод на базі ***3х-фазного асинхронного двигуна***»(*тип* ***Stop Motor***) що має наступні утворюючи складові:

1.1 – *3х-фазний асинхронний двигун* з короткозамкненим ротором;

1.2 – дві електромагнітні муфти (руху і гальмування);

1.3 – ***датчикі*** кута повороту, частоти обертання головного валу машини і датчик кута повороту педалі при натисканні на педаль носком та п'яткою ноги;

1.4 – ***контролер*** з драйверами;

1.5 – ***блок живлення***.

Модифікація ***М2*** – «Автоматизований електропривод швейних машин на базі ***крокового двигуна***», що має наступні утворюючи складові:

2.1 – ***кроковий двигун*** (***Stepper Motor***), а саме безколекторний синхронний двигун постійного струму;

2.2 – ***контролер*** з цифрової мікропроцесорної системою керування.

2.3 – ***блок живлення постійним струмом***.

Модифікація ***М3*** – «Автоматизований електропривод швейних машин на базі ***вентильного двигуна***», що має наступні утворюючи складові:

3.1 – ***вентильний двигун*** (*тип* ***BLDC*** – ***B****rush* ***L****ess* ***D****irect-****C****urrent*), а саме «безколекторний (без щіток) двигун постійного струму» зі зворотнім зв'язком;

3.2 – ***датчики*** положення ротора, які виконані у вигляді трьох фотоелектричних датчиків або у вигляді трьох датчиків Холла, які зміщені по фазі на кут 120^0 для зворотного зв'язку зі схемою керування;

3.3 – ***контролер*** і драйвер з вентильними (напівпровідниковими) ключами для автоматичного переключення котушок статора по сигналам зворотного зв'язку від датчиків положення ротора;

3.4 – ***блок живлення постійним струмом***.

Модифікація ***М4 –*** «Прямий сервопривод автоматизованих швейних машин на засадах сервопрводу» або «прямий привод типу ***DD*** – ***D****irect* ***D****rive*». В такому приводі відсутнє використання пасових і інших механічних передач. І тому привод зчеплення типу ***Clutch Motor*** для

неавтоматизованих швейних машин і привод зчеплення типу ***Efka Vario Stop*** з електромагнітними муфтами зчеплення і гальмування для автоматизованих швейних машин все більше замінюється сервоприводом, ротор якого змонтований на торці головного валу, а статор закріплений на корпусі швейної машини.

В комп'ютерних вишивальних автоматах використовуються один двигун модифікації ***M4*** (типу ***DD***) та 5 двигунів модифікації ***M2*** (типу ***Stepper Motor***) для 2D-механізму п'ялець, для каретки з голководами і ниткопритягувачами та для програмно керованих механізмів обрізки ниток і затискання кінців обрізаних ниток. В комп'ютерних швейно-вишивальних машинах використовуються один двигун модифікації ***M3*** (***BLDC***) і чотири двигуна модифікації ***M2*** (***Stepper Motor***). В комп'ютерних напівавтоматах для виготовлення петель, закріпок та для пришивання ґудзиків або іншої фурнітури використовуються один двигун модифікації ***M3*** (***BLDC***) і два двигуна модифікації ***M2*** (***Stepper Motor***). В комп'ютерних побутових швейних машинах і комп'ютерних машинах для малого бізнесу використовуються, найбільш часто, три двигуна модифікації M2 (***Stepper Motor***).

4.2. **Крововий електропривод автоматизованих машин легкої промисловості і CNC-верстатів**

Одноосні або однокординатні модулі лінійних переміщень з програмованими зворотно-поступальними рухами веденої ланки широко застосовуються у CNC-верстатах і в CNC-машинах легкої промисловості. Таки маніпулятори мають крововий електропривод з різними видами передач, які перетворюють програмовані зворотно-обертові рухи ротора крокового двигуна у лінійні рухи по одної осі 3D-координат. В структурі технологічної CNC-машини або CNC-верстата є два одноосних модуля лінійних переміщень і тому зворотно-поступальні рухи каретки (п'ялець) або столешниці відбуваються по вісям OX і OY в горизонтальної площині. Якщо в структурі верстату є три модуля лінійних переміщень, то утворюється структура 3х-координатного верстату CNC і так далі до ускладнення структури верстатів і обробляючих центрів до 5-ти і 6-координатних мехатронних систем машинобудування.

Наприклад, типовий кроковий привод механізму зсуву голочниці (фонтури) з кулько-гвинтовою передачею використовується в комп'ютерних плоско в'язальних машинах Stoll (Німеччина), ShimaSeiko (Японія) та інших

У CNC-верстатах використовується одноосні маніпуляри з кулько-гвинтовою передачею для програмованих переміщень заготовки та інструменту, наприклад, фрез по трьом, чотирьом або п'яти осям координат.

При об'єктно-орієнтованому проектуванні автоматизованих машин легкої промисловості і CNC-верстатів машинобудування на засадах крокового електроприводу і сервоприводу потрібна мати уяву про конструктивні особливості і принцип роботи та програмування таких програмуємих мехатронних виконавчих механізмів.

При підготовки студентів для роботи на посадах інженера-механіка часто виникає питання ЧИ ОБОВ'ЯЗКОВО ЗНАТИ МОВИ ПРОГРАМУВАННЯ, ЩОБ ПРОГРАМУВАТИ КОНТРОЛЕРИ на засадах мікроконтролерів для керування кроковим приводом і сервоприводом. Відповідь тут може бути наступна. Існують середовища для візуального програмування світових виробників програмуемих контролерів фірм Siemens, ABB, Schneider Electric, ОВЕН та ін. Але всі ці середовища програмування прив'язані до контролерів промислової автоматики і до певного виробника та мають високу вартість.

Програмне середовище і плати **Arduino** ідеально підходять для вивчення програмування контролерів студентами-механіками. Ці плати програмуються на спрощеної об'єктно-орієнтованої мові програмування **Arduino C**. Для поглибленого вивчення і застосування об'єктно-орієнтованих мов програмування контролерів запрошуються професійні програмісти. Для програмування контролерів без знання основ мови програмування **Arduino C** поширене застосування **FLProg** - системи візуального програмування плат Arduino. Всі контролери автоматизованих машин побудовані на мікроконтролерах і за допомогою драйверів підключаються до датчикив і виконавчих механізмів зворотно-поступової та обертової дії цільового призначення. Для роботи зовнішніх пристроїв програма при завантаженні її в контролер *компілюється.*

Компіляція це «переклад» програмою-компілятором з мови високого рівня, яку склала і завантажила в комп'ютер людина на мову машинних кодів (нулів і одиниць), яку розуміє комп'ютер.

Інженеру-механіку НЕ ОБОВ'ЯЗКОВО ЗНАТИ МОВИ ПРОГРАМУВАННЯ високого рівня, ЩОБ ПРОГРАМУВАТИ КОНТРОЛЕРИ на засадах мікроконтролерів. Програма **FLProg** дозволяє створювати прошивки для плат Arduino (програмування контролера) на графічної мові ***FBD*** *(**F**unction **B**lock **D**iagram* – мова функціональних блоків*)* або на графічної мові ***LD*** *(**L**edder **D**iagram* – мова релейних схем*)* інструментального програмного комплексу промислової автоматизації ***CoDeSys*** *(**Co**ntroller **De**velopment **Sys**tem)* міжнародного стандарту **МЕК 31131-3.**

Кроковий електропривод і сервопривод сучасних технологічних CNC-машин галузі, CNC-верстатів машинобудування, робототехнічних і мехатронних засобах автоматизації механіко-технологічних систем будується на засадах крокових двигунів (**КД**). Кроковий двигун це безколекторний двигун постійного струму, який містить котушки на статорі і постійні магніти на зубчастому роторі. І тому поширена назва крокових двигунів «крокові двигуни з постійними магнітами». За способом живлення котушок на статорі на практиці найбільш поширені наступні: КД «**біполярні крокові двигуни**», що мають по дві котушки на кожну фазу та «**уніполярні** (одно полярні) **крокові двигуни**», що мають по одної котушки на кожну фазу. Перемикання котушок виконується автоматично за програмою контролера на засадах мікроконтролера. При імпульсному збудження (перемиканні) котушок крок за кроком повертається вектор загального магнітного поля, яке утворюється магнітним полем котушок статора і магнітним полем постійних магнітів ротора.

На відміну від асинхронних двигунів змінного або колекторних двигунів постійного струму крокові двигуни управляються цифровими імпульсами. Імпульсний код на виході програмуємого контролера перетворюється в малий кут (крок) повороту ротора. Як і всі двигуни, крокові двигуни складаються зі статора і ротора. На рис. 4.1,а наведена спрощена схема крокового двигуна. На роторі встановлені постійні магніти, а до складу статора входять 4 котушки (обмотки) обмотки, які розташовані по колу під кутом 90°.

Відмінності в способах підключення обмоток визначають режим роботи крокового двигуна. Напрямок обертання валу визначається порядком, в якому з'єднані котушки між собою.

ТИПИ КРОКОВИХ ДВИГУНІВ.

Розглянемо найбільш поширені крокові двигуни, які покладені в основу проектування крокового приводу (кроковий двигун + передача +

система керування) технологічних машин легкої промисловості і верстатів машинобудування.

КРОКОВИЙ ДВИГУН З ПОСТІЙНИМ МАГНІТОМ (рис. 4.3). Ротор такого двигуна містить постійний магніт у формі диска з двома або більшою кількістю полюсів. Котушки із сердечниками статора утворюють по колу електромагніти, які при живленні їх імпульсними електричними сигналами притягувати та/або відштовхувати відповідні протилежні полюси (зубці) постійний магніт на роторі і створювати тим самим крутний момент. На рис. 4.3 наведена принципова схема крокової двигуна з постійним магнітом.

а б

Рис. 4.3. Принципові конструктивні схеми крокових двигунів: а – з ротором у вигляді постійного магніту; б - з ротором у вигляді зубчастого диска з магнітно м’якого металу без постійного магніту

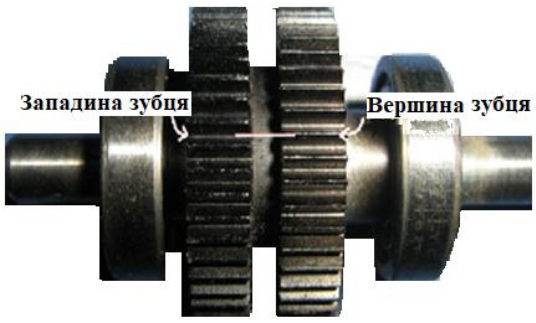
КРОКОВИЙ ДВИГУН ЗІ ЗМІННИМ МАГНІТНИМ ОПОРОМ (рис. 4.3,б). У двигунів цього типу на роторі немає постійного магніту. Замість цього, ротор виготовляється з магнітно м’якого металу у вигляді шестерінки. Статор має більше чотирьох обмоток. Обмотки живляться в протилежних парах і притягують ротор. Відсутність постійного магніту негативно впливає на величину крутного моменту, він значно знижується. Але у цих двигунів немає стопового (утримуючого) моменту. Стоп-момент це момент, що обертає ротор при наявності струму в котушках але при відсутності струму в котушках постійні магніти ротора стопоряться взаємодією з феромагнітними сердечниками статора. Стоп-момент можна відчути якщо спробувати повернути рукою відключений кроковий двигун з постійним магнітом. Ви відчуєте помітні клацання на кожному кроці двигуна. Насправді те, що ви відчуєте і буде фіксуючим моментом, який притягує магніти до «заліза» статора. Крокові двигуни зі змінним

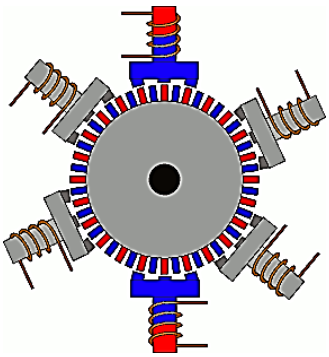
магнітним опором зазвичай мають крок, що лежить в діапазоні 5°…15°.

ГІБРИДНИЙ КРОКОВИЙ ДВИГУН (рис. 4.4)**.** Назву «гібридний» кроковий двигун отримав через те, що він поєднує в собі характеристики крокових двигунів з постійними магнітами і характеристики крокових двигунів з перемінним магнітним опором (рис. 4.4)**.**

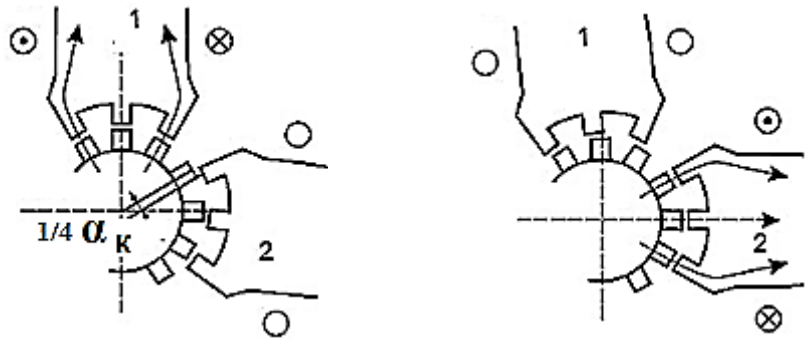
Гібридні крокові двигуни мають високій утримуючий і динамічний крутний моменти, а також малу величину кроку, що лежить в межах 0.9…5 градусів, що забезпечує високу точність позиціювання ведених ланок програмно керованих механізмів технологічних машин і верстатів машинобудування. Цей тип двигунів широко використовується в CNC-верстатах, роботах і мехатронних модулях технологічних машин галузі.

Поширений у використанні на виробництві гібридний кроковий двигун з 8-ю обмотками (4-ма парами), який виконує 200 кроків на один оборот. Ротор такого гібридного крокового двигуна має 50 позитивних і 50 негативних зубців-полюсів. Через те, що такий постійний магніт важко зробити, було знайдено оригінальне технічне рішення. Ротор містить два окремих 50-зубчастих диска (рис.4.4,а). Також використовується циліндричний постійний магніт закріплений на валу ротора. Перший зубчастий диск закріплений з позитивним полюсом постійного магніту, а другий зубчастий диск закріплений до негативного полюсів постійного магніту. Таким чином, один диск має позитивний полюс (S) на своїх 50 зубцях, а другий має негативний полюс (N) на своїх 50 зубцях. При цьому зубці одного диску зміщені на 1 зуб відносно зубців другого диску. І тому западини зубців на одному диски розташовані проти вершин зубців на другому диску. В результаті такого конструктивного рішення отримано ротор із 100 зубцями, де кожний зуб це полюси (S) і (N) постійного магніту, який розташований всередині зубчастих двох дисків і ці полюси (S) і (N) зубців ротора чергуються по колу (рис. 4.4,б).





а б



в г

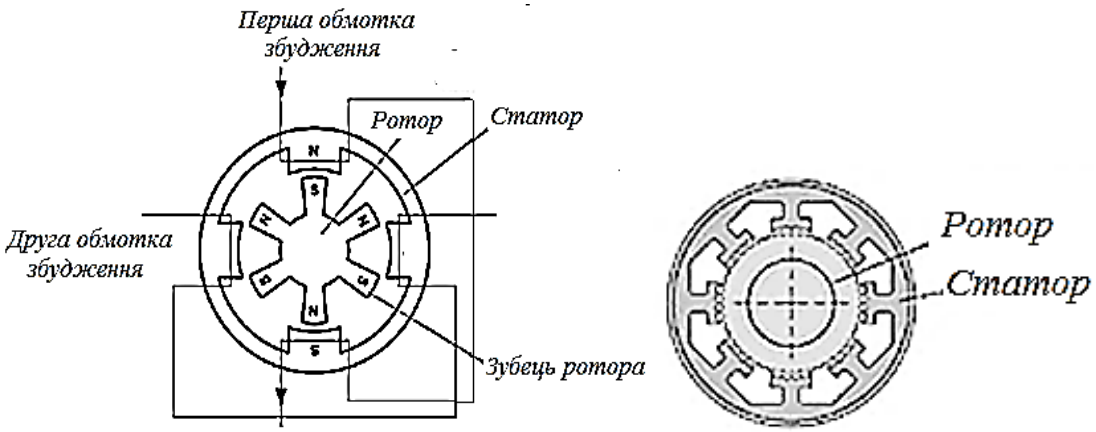
Рис. 4.4. Конструктивна схема ротора гібрідного крокового двигуна (а), збіг вектора магнітного поля зубців постійних магнітів ротора з вектором магнітного поля активних електромагнітів статора в початковому положенні ротора (б, в) та стійка магнітна рівновага ротора через один крок (г)

На рис. 4.4,в котушки розташовані на статорі під кутом в 60°, але якщо припустити, що перша пара - це сама верхня і сама нижня котушки, тоді друга пара зміщена під кутом 60° + 5° (рис. 4.4,г) по відношенню до першої, і третя зміщена на 60 + 5 ° по відношенню до другої. Ця кутова різниця і є причиною обертання ротора. Режими керування з повним і половинним кроком можуть використовуватися для зниження енергоспоживання.

На рис. 4.5,а зображена конструкція 2-фазного крокового двигуна, який має 6 зубців ротора (крок 15^0 при повнокроковому режимі керування), а на рис 4.5,б – 50 зубців ротора (крок $1{,}8^0$ при повнокроковому режимі керування).

Кроковий двигун побудований таким чином, що деякі его характеристики можна прийняти за неполадки, якщо не знати їх причини. Такі приводи часто нагріваються більше ніж приводи інших типів тому, що через обмотки статора крокового двигуна завжди, навіть коли ротор нерухомий проходить повний робочий струм, який створює так званий «**утримуючий момент**». Крутний момент, створений двигуном повинен подолати утримуючий момент і бути більше моменту з боку навантаження крокового приводу з боку технологічної машини. Останній складається з приведених до валу ротора моментів інерції механізмів технологічної

машини, моментів опору від сил тертя та моментів від сил корисного опору при обробці матеріалів робочими інструментами. Котушки крокових двигунів мають ізоляцію класу «Н». Це означає, що температура обмоток може збільшуватися до 120^0. Нагрівання корпусу статора до 80^0 це нормальне явище.



а б

Рис. 4.5. Конструктивні схеми крокових двигунів: а – 24 кроків на 1 оборот ротора з 6 зубцями (постійними магнітами); б – 200 кроків на 1 оборот ротора з 50 зубцями (гібридний)

Кроковий двигун не може працювати без навантаження. Для нормального прискорення йому потрібен протидіючий момент інерції, який буде не менше ніж момент інерції ротора. Без такого навантаження кроковий двигун буде працювати в рознос.

При включенні кроковий двигун може смикнутись. Це пов'язано з тим, що він має фіксоване число положень (*кроків*) на один оборот ротора. В крововому двигуні серія електричних імпульсів перетворюється в обертання ротора по крокам за стрілкою годинника або проти стрілки годинника. Один код серії електричних імпульсів забезпечує поворот ротора на один певний кут (*один крок*) у відповідності з наступної формулою:

$$\textbf{1 крок} = \frac{\mathbf{360^0}}{\mathbf{2}\cdot(\text{кількість фаз})\cdot(\text{кількість зубців ротора})}$$

Наприклад, для крокового двигуна на рис. 4.3,а один крок складає $(360^0/2 \cdot 2 \cdot 6) = \pm 15^0$, а для крокового двигуна на рис. 4.3,б один крок складає $(360^0/2 \cdot 4 \cdot 50 = \pm 0{,}9^0 = \pm 54^/$.

Підключення крокових двигунів потрібна виконувати екранованими кабелями зі скрученими проводами (жилами) типу кручена пара (*рос. витая пара*). Екран кабелю потрібна заземлювати з обох кінців. Екранування дозволяє виключити проникнення в систему електромагнітних перешкод, а також їх випромінюванню системою керування.

Принцип роботи пояснюється схемами на рис. 4.6. Кожна фаза збудження має дві полюсні котушки на статорі, які при проходженні по ним імпульсу струму позитивної або негативної полярності створює різнойменні полюса на статорі.

Всі крокові приводи мають розглянутий принцип дії. Відмінності лише в тому, що у двигуна з більшою кількістю полюсів статора і зубців ротора ротор за один оберт робить більше кроків але меншого розміру.

При проходженні струму по котушкам 1-2 (рис. 4.6,а) першої фази збудження відповідні полюса статора намагнічуються і притягають зубці ротора протилежної полярності. При підключенні котушок 1-2 і 3-4 двох фаз збудження (рис. 4.6,б) ротор повертається ще на кут 15^0. Для наступного кроку відбувається відключення котушок 1-2 першої фази, а залишається підключеними тільки котушки 3-4 і тому ротор повертається ще на кут 15^0(рис. (рис. 4.6,в). А далі відбуваються наступні переключення котушок: 3-4 та 2-1, 2-1, 2-1 та 4-3 і так далі. Для повного одного оборот ротора за стрілкою годинника на рис. 4.4 потрібно було навести 24 положення ($15^0 \cdot 24 = 360^0$). Таким чином, робота 2х-фазного крокового двигуна (рис. 4.6) в навпіл кроковому режимі (рис. 4.7,б) відбувається за наступним алгоритмом:

1-2 (рис. 4.6,а – перший крок) → **1-2, 3-4** (рис. 4.6,б – другий крок) → **3-4** (рис. 4.6,в – третій крок) →**3-4, 2-1** (рис. 4.6,г – четвертий крок) → **2-1** (рис. 4.6,д – п’ятий крок) → **2-1, 4-3** (рис. 4.6,е – шостий крок) і так далі.

Для 3-фазного крокового двигуна з 4 зубцями ротора кількість кроків за 1 оберт ротора складає 24, що розраховується за формулою:

2·(число фаз = 3)·(число зубців ротора = 4) = 2·3·4 =24.

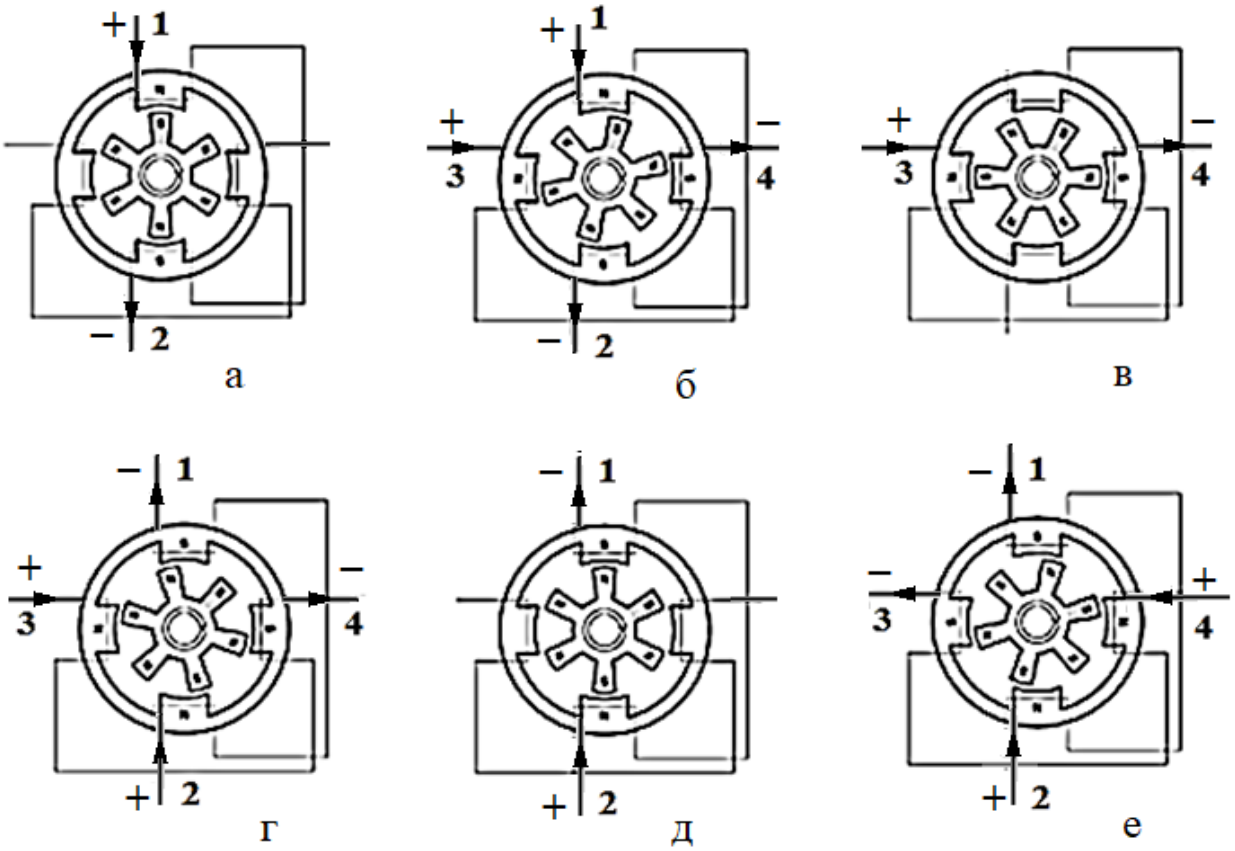
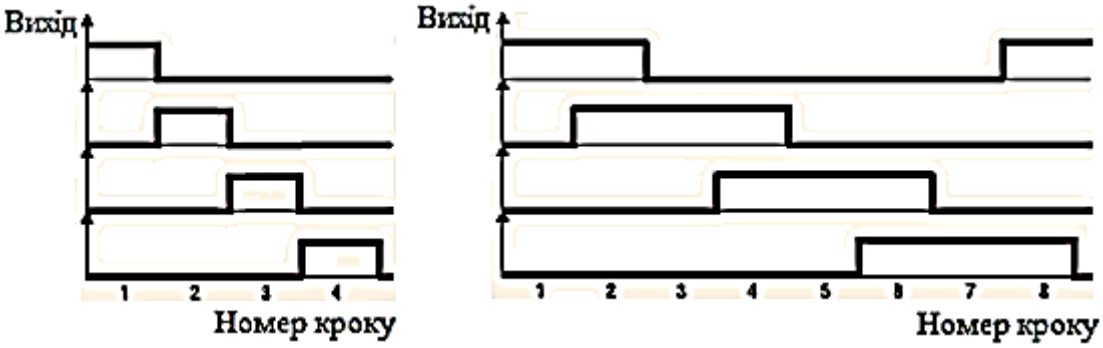


Рис. 4.6. Схеми для 6 кроків крокового двигуна, який має 4 котушки на статорі та 6 зубців на роторі, утворених трьома постійними магнітами



а б

Рис. 4.7. Діаграми подачі імпульсів на обмотки крокового двигуна: а – повно кроковий режим керування; б – навпіл кроковий режим керування

Кут повороту ротора на 1 крок складає $\frac{360^0}{24} = 15^0$.

Для двофазного крокового двигуна з 6 зубцями ротора кількість кроків за 1 оборот ротора складає також 24 і розраховується за формулою:

2·(число фаз = 2)·(число зубців ротора = 6) = 2·2·6 = 24. Кут повороту ротора на 1 крок складає $\frac{360^0}{24} = 15^0$.

Частота обертання ротора визначається частотою електричних імпульсів, тобто швидкістю переключень обмоток збудження.

РЕЖИМИ РОБОТИ КРОКОВИХ ДВИГУНІВ.

Розглянемо найбільш поширені чотири варіанта режимів подачі сигналів керування у вигляді електричних імпульсів постійного струму на котушки (обмотки) статора для обертання ротора у вигляді постійного магніту.

ПОЛНО КРОКОВИЙ РЕЖИМ КЕРУВАННЯ (рис. 4.8 і рис. 4.9). Для реалізації цього режиму, напруга на обмотки подається попарно. Залежно від способу підключення обмоток (послідовно або паралельно). В цьому випадку двигун буде видавати 100% номінального крутного моменту.

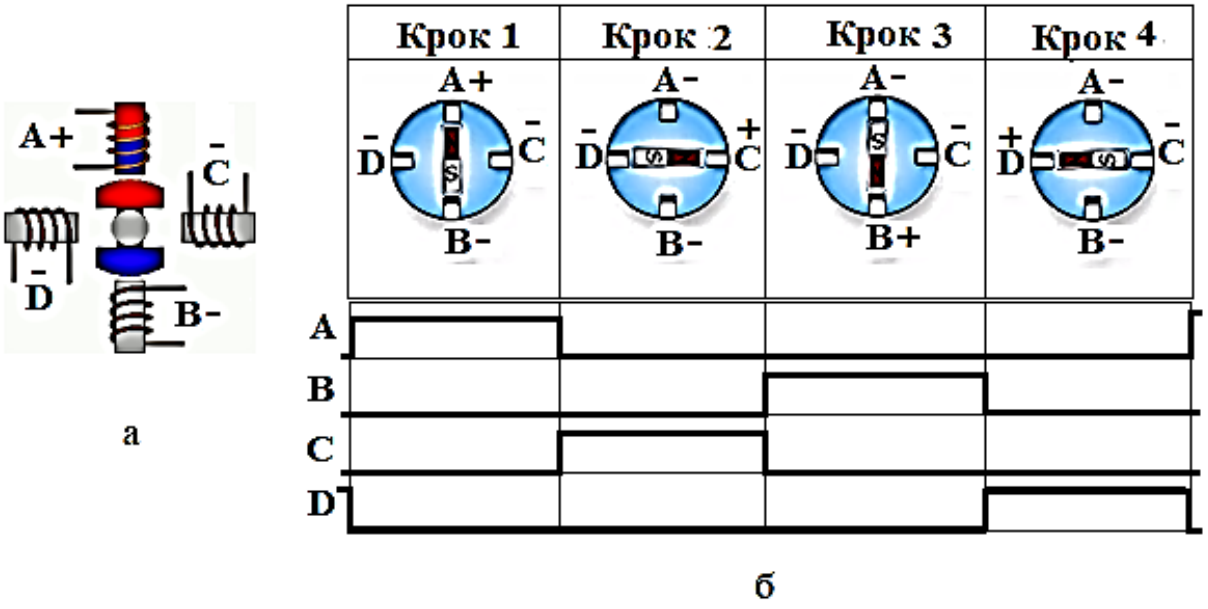


Рис. 4.8. Повно кроковий режим роботи біполярного крокового двигуна (**варіант 1**): а – живлення котушки А для кроку 1; б – положення ротора (постійних магнітів) для чотирьох тактів (чотири кроку) A → C → B → D програмного почергового перемикання по одній фазі статора для одного циклу обороту ротора

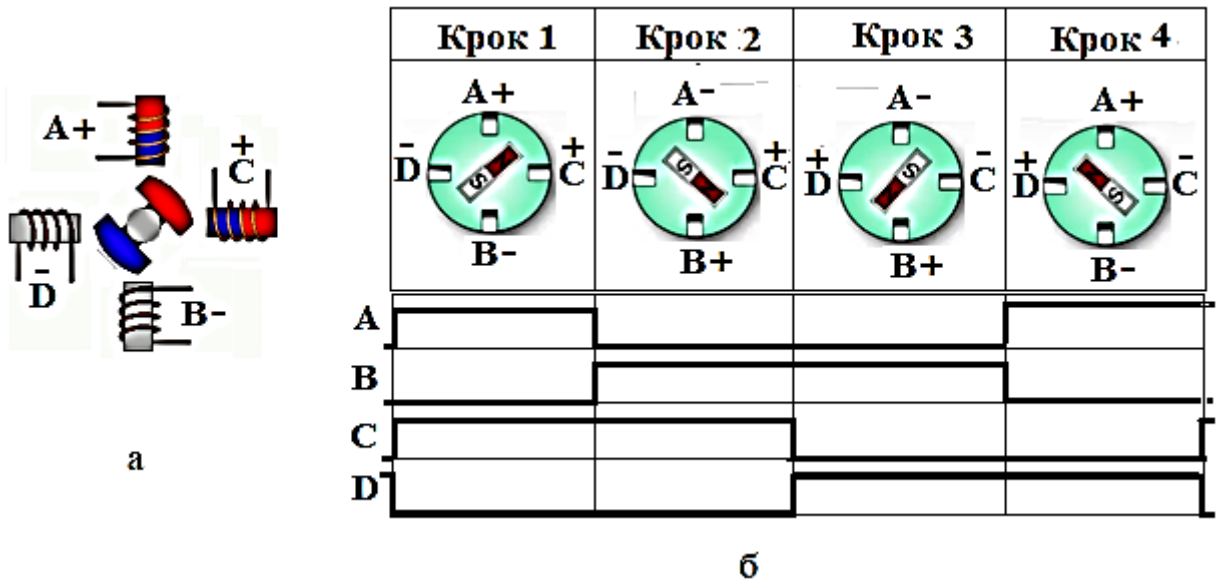


Рис. 4.9. Повно кроковий режим роботи біполярного крокового двигуна (**варіант 2**): а – живлення котушок А та С для крока 1; б – положення ротора (постійних магнітів) для чотирьох тактів (чотири кроку) АС → СВ → BD → DA програмного почергового перемикання по дві фази статора для одного циклу обороту ротора

НАВПІЛ КРОКОВИЙ РЕЖИМ (**рис. 4.10**). Не змінюючи при нічого в «залізі» двигуна для отримання подвоєної точності позиціонування валу ротора порівняно з повно кроковим режимами керування застосовується навпіл кроковий режим керування кроковим двигуном (**варіант 3**). Для реалізації цього режиму одна пара обмоток або дві пари обмоток програмно отримують живлення одночасно і тому ротор повернеться на половину крок порівняно з варіантам 1 варіантом 2 повно крокових режимів керування. При цьому кроковий двигун виконує подвійну кількість кроків на оборот, що означає подвійну точність для системи позиціонування.

Навпіл кроковий режим роботи біполярного крокового двигуна (**варіант 3**) наведений на рис.4.10, де зображені положення постійного магніту ротора і 8 тактів **AB → AB+CD → CD → CD+BA → BA → BA+DC → DC → DC+DA** програмного почергового перемикання фаз статора для одного циклу обороту ротора.

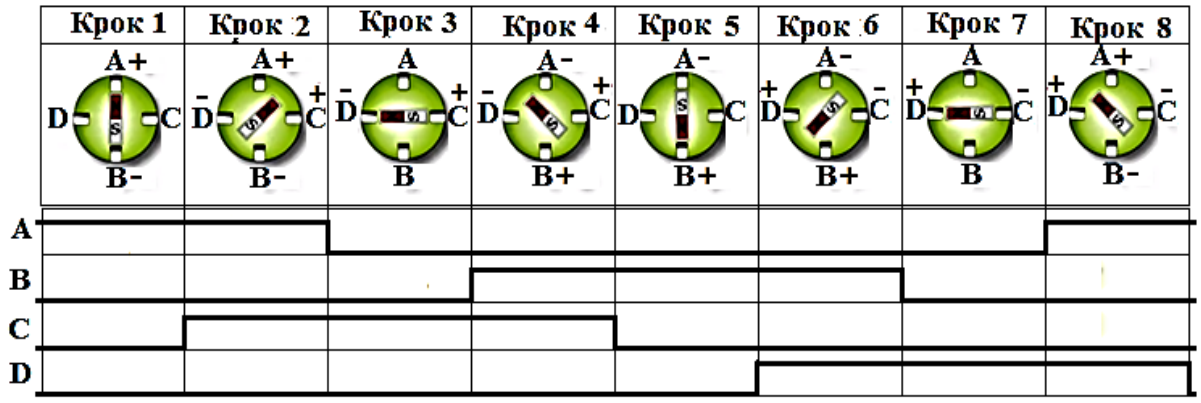
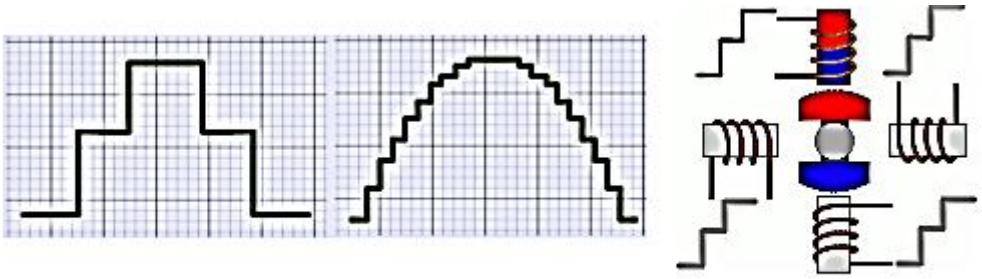


Рис. 4.10. Навпіл кроковий режим роботи біполярного крокового двигуна (**варіант 3**) положення постійного магніту ротора для 8 тактів (8 кроків) програмного почергового перемикання по дві фази статора для одного циклу обороту ротора

РЕЖИМ МІКРОКРОКА (**рис. 4.11**). Для підвищення рівня плавності ходу крокових двигунів крім навпіл крокового режиму роботи застосовуються мікрокроковий режим (**варіант 4**) і режим ШІМ регулювання. Ідея мікрокроку полягає в подачі на обмотки двигуна живлення не імпульсами, а цифровим сигналом (рис.4.11,а), який за своєю формою нагадує синусоїду. При цьому відбувається більш гладке переміщення ротора при переходах від одного кроку до наступного, що визначає широке використання мікрокрокового режиму керування в системи позиціонування CNC-верстатах (верстатах з ЧПК). Крім цього значно зніжуються крокові прискорення і навантаження в системі «кроковий двигун – технологічна машина».

На рис.4.11 наведені приклади апроксимації полуперіода синусоїди цифровим сигналом за допомогою **АЦП** (**А**налого-**Ц**ифового-**П**еретворювача) контролера крокового двигуна:



а б в

Рис. 4.11. Результат аналого-цифрового перетворення сигналу керування для режиму мікрокроку з різним дозволом (а, б) і принципова схема живлення котушок крокового двигуна (в)

В швейних циклових напівавтоматах, в швейно-вишивальних машинах і вишивальних CNC-автоматах із зубчасте-пасовою передачею використовується два одноосних програмно керованих механізма (ПКМ) лінійних переміщень по осям координат OX і OY із затиснутим в п'яльцях чи касеті текстильним матеріалом відносно голки, човника і притискної лапки.

Сумарна податливість і значення маса-інерційних параметрів динамічної системи програмно-керованих ПКМ траверси $\boldsymbol{m}_{\text{т}}^{*}$ більше аналогічних параметрів ПКМ каретки $\boldsymbol{m}_{\text{к}}^{*}$. Закони керування моментами кроковими двигунами $\text{М}_{\text{КД}-\text{X}}$ і $\text{М}_{\text{КД}-Y}$ реалізуються програмно. Типовий алгоритм формування $\text{М}_{\text{КД}-\text{X}}$ і $\text{М}_{\text{КД}-Y}$ ПКМ циклової швейної CNC-машини, таблиці двійково-вісімкових кодів «2»–«8» і десяткових кодів наведено на рис. 4.12. Для двох варіантів програмної комутації крокових приводив **КД-X** і **КД-Y** (3-фазних з 6 котушками збудження на статорі при 12-тактної комутації) для алгоритму (рис. 4.12,а) прийняти наступні умовні позначення:

N=12 – число тактів комутації КД-X і КД-Y;

$\{\text{a}_{\boldsymbol{i}}\}$ – таблиця (рис. 4.12,б і рис. 4.12,в) кодів комутації, де $i = \overline{1,n}$ – номер комутації;

$\boldsymbol{x_i, y_i}$ – число тактів комутації в кроках лінійних переміщень програмуємої довжини машинних стібків;

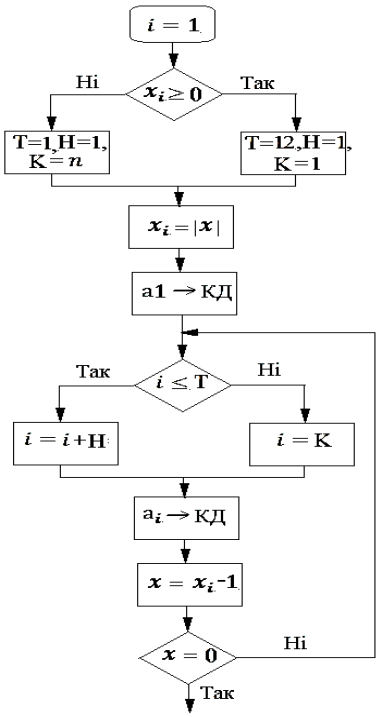
T, H, K – проміжні постійні.

Напрям обертання ротора КД за стрілкою годинника програмується наступною послідовністю покроковою комутацією $\text{a}_{\mathbf{1}}$ …$\text{a}_{\mathbf{12}}$ шості котушок статора через одну-дві котушки для каретки з меншою приведеною масою $\boldsymbol{m}_{\text{к}}^{*}$:

→1→1,2→2→2,3→3→3,4→4→4,5→5→5,6→6→6,1→

та через дві-три котушки для траверси з більшою приведеною масою $\boldsymbol{m}_{\text{т}}^{*}$:

→1,2→1,2,3→2,3→2,3,4→3,4→3,4,5→4,5→4,5,6→5,6→5,6,1→6,1→6,1,2→



а

{a$_i$}	К о д	
	«2» - «8»	«10»
a_1	110 000	60
a_2	111 000	70
a_3	011 000	30
a_4	011 100	34
a_5	001 100	14
a_6	001 110	16
a_7	000 110	6
a_8	000 111	7
a_9	000 011	3
a_{10}	100 011	43
a_{11}	100 001	41
a_{12}	110 001	61

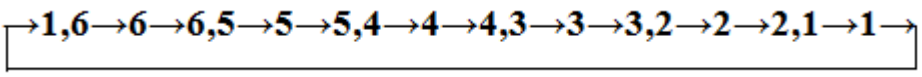
б

{a$_i$}	К о д	
	«2» - «8»	«10»
a_1	100 000	40
a_2	110 000	60
a_3	010 000	20
a_4	011 000	30
a_5	001 000	10
a_6	001 100	14
a_7	000 100	4
a_8	000 110	6
a_9	000 010	2
a_{10}	000 011	3
a_{11}	000 001	1
a_{12}	100 001	41

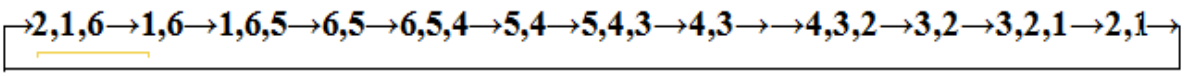
в

Рис. 4.12. Блок-схема алгоритму (рис. а) формування програмуємих керуючих впливів на $М_{КД-Х}$ і $М_{КД-У}$ ПКМ двокоординатних швейних CNC-машини; таблиця кодів тактової комутації крокового двигуна КД-Х (рис. б) таблиця кодів тактової комутації крокового двигуна КД-У (рис. в)

Напрям обертання ротора КД проти стрілки годинника програмується наступною послідовністю покроковою комутацією a_{12} …a_1 шості котушок статора через одну-дві котушки для каретки з меншою приведеною масою $m_к^*$:



та через дві-три котушки для траверси з більшою приведеною масою $m_т^*$:



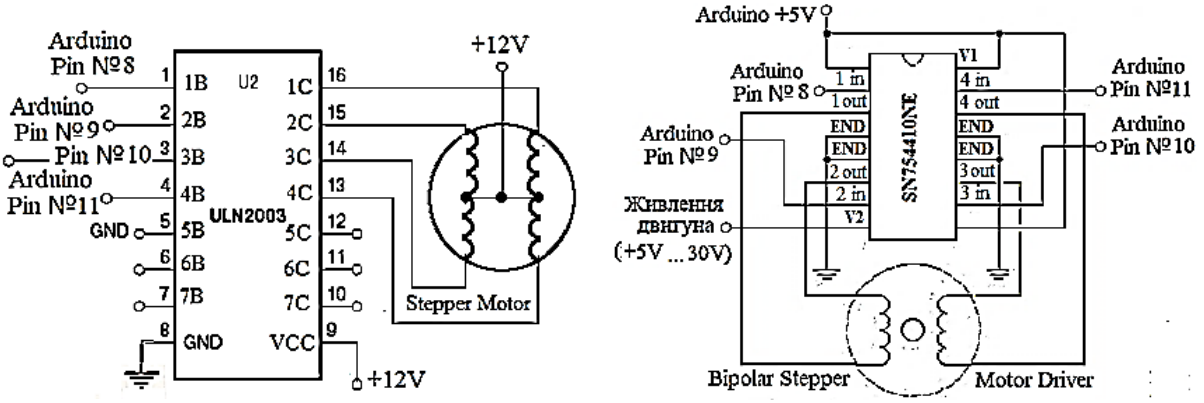
Програмуемий контролер, драйвер, кроковий двигун і блок живлення це основні складові крокового приводу. Послідовності електричних імпульсів поступають програмно від контролера до драйвера крокового двигуна в обмотки збудження з деяким зсувом в часі і утворюється вектор магнітного поля статора. Постійні магніти статора створюють свій вектор магнітного поля. Коли вектор магнітного поля статора співпадає з вектором магнітного поля ротора утворюється вектор спільного магнітного поля системи «статор – ротор» і ротор повертається на один крок. За допомогою ШІМ-регулювання шпаруватістю імпульсів контролер регулює швидкість крокового двигуна і кількість кроків / навпіл кроків або мікрокроків двигуна, а значить і функції положення, швидкості і прискорення веденої ланки програмно керованих механізмів технологічної машини.

Пропуск імпульсів (кроків) при роботі крокового двигуна може бути як з механічних причин, наприклад за рахунок резонансних коливань в механічних з'єднаннях між приводом і зовнішнім навантаженням, також по причині збою роботи електронних частин блока керування, а також по причині відсутності зворотного зв'язку в структурі блока керування. Вибір для технологічної машини або для механізму крокового двигуна заниженої потужності також приводить до пропуску механічного кроку тому, що такий двигун не може подолати зовнішнє навантаження при заданому прискоренні і частоти обертання.

Якщо зовнішнє навантаження не підключено або мале за величиною кроковий двигун може працювати в рознос.

В залежності від конструкції (**біполярний** або **уніполярний**) і способу (схеми) підключення крокового двигуна застосовують різні драйвери. Наприклад, для поширених контролерів **Arduino UNO** (рис. 4.2,а) на рис. 4.13,а наведена схема підключення уніполярного крокового двигуна **(**4

виходи**)** за допомогою мікросхеми **ULN2003** однойменного драйвера, а на рис. 4.13,б – схема підключення біполярного крокового двигуна **(**4 виходи**)** за допомогою мікросхеми **SN754410NE** однойменного драйвера.

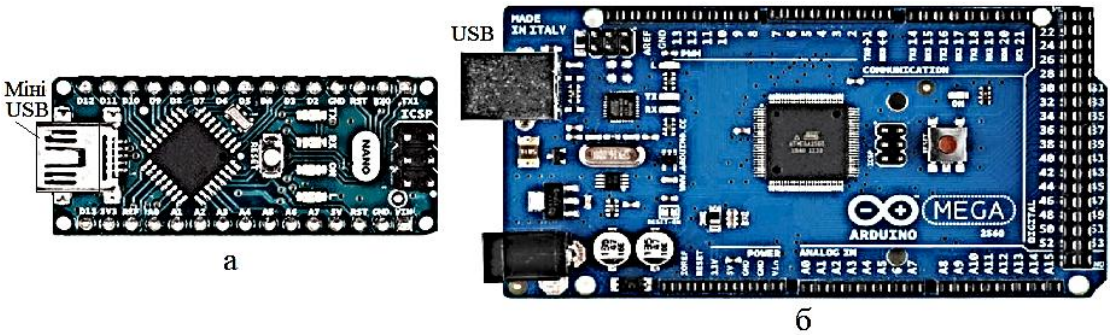


а б

Рис. 4.13. Схема підключення уніполярного крокових двигунів: а – уніполярного (**Unipolar Stepper Motor**): б – біполярного (**Biipolar Stepper Motor**)

4.3. **Платформа і контролер Arduino UNO**

У користувачів має перевагу у використання плати контролера **Arduino UNO**, яка містиць мікроконтролер ATmega **328** (**32** - 32 Кбайт флеш-пам’ять. **8** – 8-розрядний) Поширені також моделі плати контролерів (рис.4.14) **Arduino Nano** (встановлюється на макетної платі) і **Arduino Mega** з мікроконтролером ATmega **2560**, що має 54 цифових пінів проти 14 у Arduino UNO і 16 аналогових пінів проти 6 у Arduino UNO та 4 COM-портів проти одного у Arduino UNO.

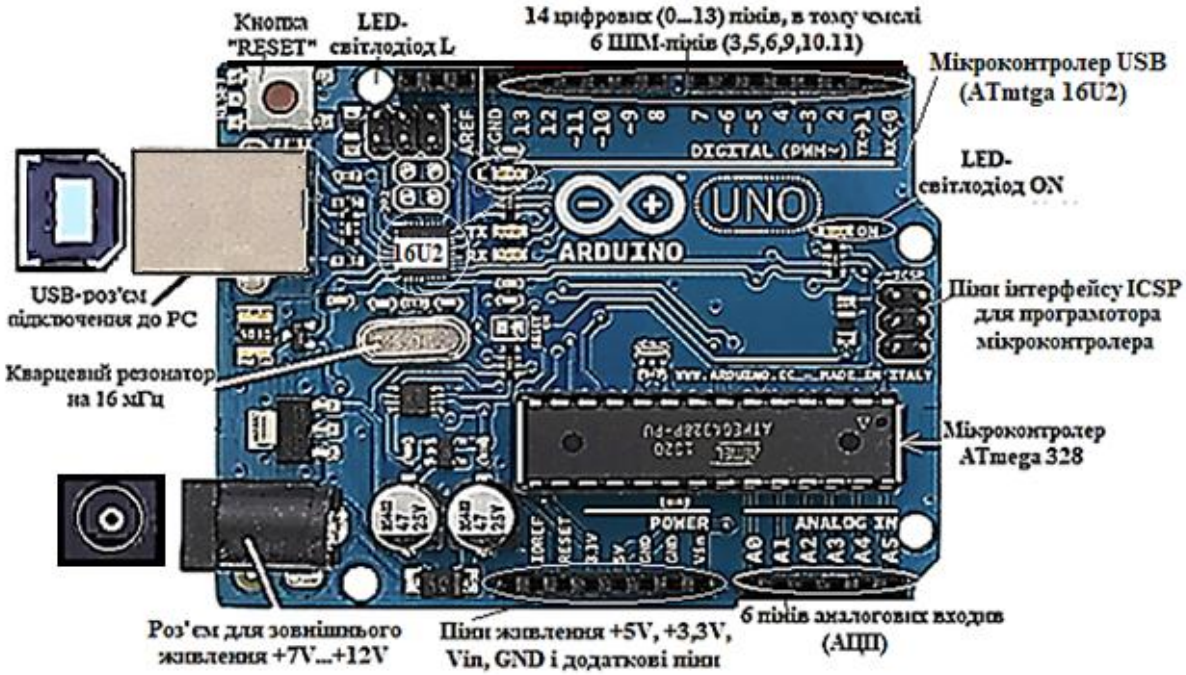


а б

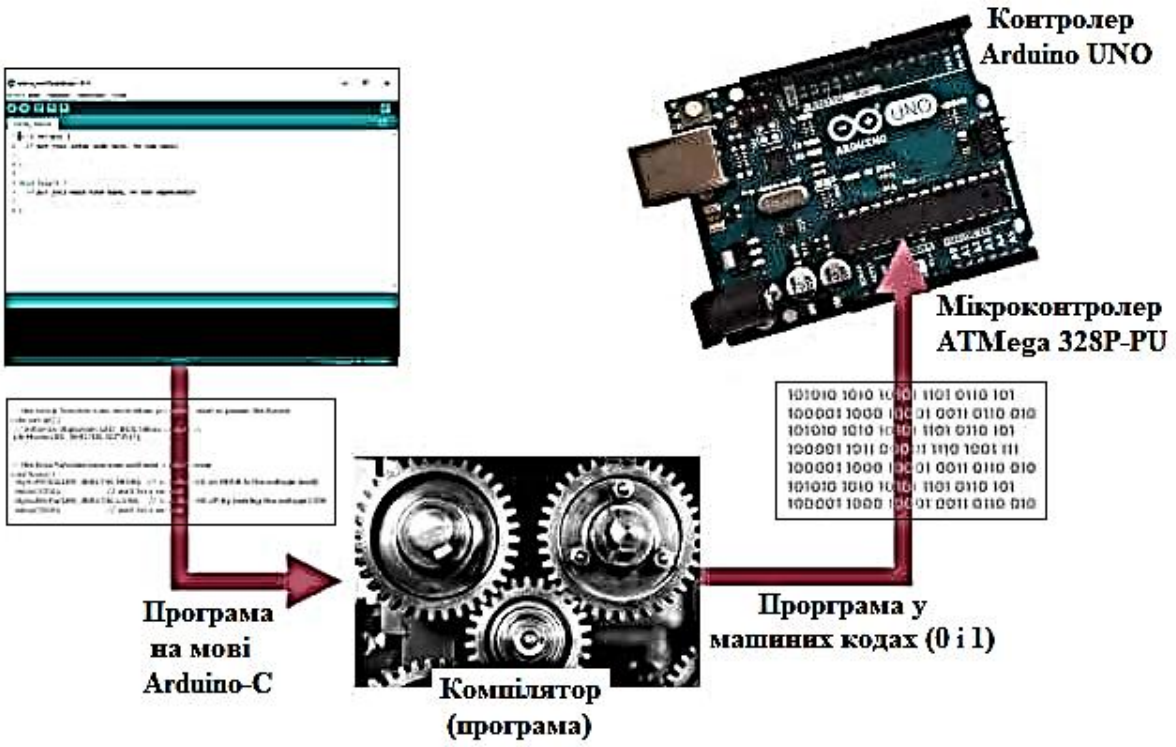
Рис. 4.14. Загальний вигляд контролерів **Arduino Nano** (а) і **Arduino Mega** (б)

Контролери Arduino (рис.4.15,а) не мають операційної системи, а працюють з інтегрованим середовищем **Arduino IDE**, у якому виконується кодування і завантаження проектів для контролерів Arduino і його багатьох клонів. Поширений також і функціонально адекватний вдосконалений варіант інтегрованого середовища розробки Arduino відомий під назвою **MariaMole IDE**

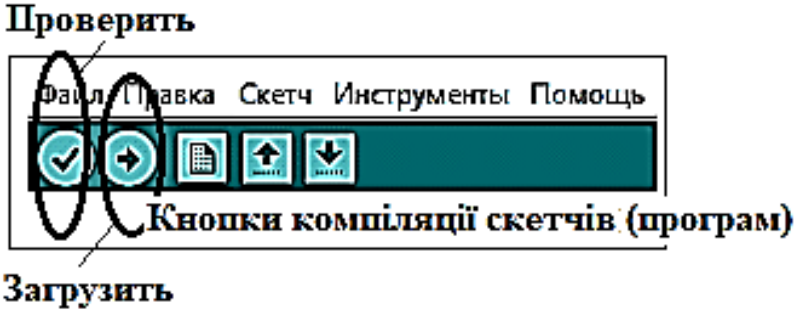
Цифрові мехатронні системи керування автоматизованими технологічними машинами і верстатами машинобудування засновані на застосуванні мікроконтролерів, які є основою побудови програмуємих контролерів або проста контролерів. Контролер за архітектурою схожий на мікрокомп'ютер, що містить мікроконтролер (**MCU**). Мікроконтролер **MCU** (**M**icro**C**ontroller **U**nit) це програмуєма багато клемна інтегральна схема, яка може виконувати команди, що зберігаються в пам'яті. Контролери працюють з периферійними пристроями, які орієнтовані на виконання **конкретних задач** мехатроніки і робототехніки за допомогою механізмів і апаратів, реалізуючих механічні і хімічні технології. Міні-комп'ютери (наприклад, *Raspberry Pi*) на відміну від контролерів орієнтовані на виконання **задач загального призначення** (розрахунків, моделювання, прогнозування, оптимізації тощо) і для цього містять



а



б



в

Рис. 4.15. Загальний вигляд і призначення основних компонентів контролера **Arduino UNO** (рис. а), схема принципового інформаційного зв'язку контролера з компілятором при компіляції програм (рис. б) та фрагмент меню (рис. в)

центральний процесор **CPU** (**C**entral **P**rocessing **U**nit), оперативну пам'ять (**ОЗУ**), пам'ять програм (**Flash-пам'ять**)і зовнішні (периферійні) пристрої у вигляді дисплею, клавіатури, принтера та інших стандартних пристроїв. Для вивчення структури і основ програмування при підготовці майбутніх інженерів-механіків доцільно починати з отриманням компетенцій з роботи поширеної платформи Arduino IDE.

Загальний вигляд і призначення основних компонентів контролера Arduino UNO наведені на рис. 4.15,а.

Контролери не розуміють програми на мовах високого рівні. Потрібна **компіляція** таких програми. *Компіляція* це переклад з мови високого рівня, наприклад з мови **Arduino-C** на мову, яку розуміє контролер (або комп'ютер) з визначенням помилок для редагування програми. Після компіляції контролер отримує програму в машинних кодах або мову нулів і одиниць. Робота компілятора для програм на мові Arduino-C для контролерів серії Arduino наочно зображена на рис. 4.15,б.

Для компіляція програми треба натиснути у меню одну з двох кнопок (рис. 4.15, в). Після компіляції програми звертається увага на поля повідомлення (статус компіляції) знизу вікна програми. Для цього виконується **перевірки програми** при компіляції (перша зліва кнопка на рис. 4.15,в) або виконується **завантаження програми** для компіляції (друга зліва кнопка). Повідомлення про помилку у програмі дублюється рожевим рядком, який підсвічується у відповідному рядку тексту програми. Поширене повідомлення помилок «не правильний синтаксис» (bad syntax), наприклад, в кінці операнду пропущена точка з крапкою або в програмі є непарна кількість фігурних дужок та інші. Якщо є декілька синтаксичних помилок у тексті програми, то вони можуть походити від попередніх. Тому потрібно спочатку виправити першу помилку синтаксису і перекомпілювати програму. При цьому інші помилки можуть автоматично зникнути.

При вивченні платформи Arduino потрібні три основних компонента:

- основна плата Arduino, наприклад, **Arduino UNO** ;
- плати і модулі розширення (драйвери, шилди, датчики, двигуни, реле та інші програмно керовані пристроїв мехатроніки та робототехніки цільового призначення);
- інтегроване середовище розробки **Arduino IDE** (**I**ntegrated **D**evelopment **E**nvironment) – система програмних засобів, яка служить для розробки і компіляції програм.

Плата контролера Arduino UNO, як і в більшість інших плат контролерів родини Arduino (рис.4.16) побудовані на засадах мікроконтролерів **Atmel**. Саме плата контролера Arduino UNO (рис.4.15,а) побудована на засадах мікроконтролера **ATmega 328** (**8**-бітовий **AVR** мікроконтролер, **32** кілобайта флеш-пам'ять). Цей мікроконтролер виконує увесь скомпільований код програми на мові **Arduino-C** і надає доступ до наступних трьох слотів (груп) пінів для підключення сигналів

вводу/виводу і периферійних пристроїв. На платі всі ці порти виведені на піни (штиркові контакти).

Перша група пінів це **14 цифрових** (**0…13**) пінів вводу/виводу, з яких одночасно піни **3, 5, 6, 9, 10** і **11** з позначкою **~** використовуються в схемах мехатронвки для ШІМ (**PWM**) сигналів, які програмно перетворюються з цифрових у псевдо-цифрові. **15ий** пін цієї групи це пін **AREF** для опорної напруги аналогових входів комунікаційних шин і для шин послідовних інтерфейсів (шіни типу **I²C** та шини типу **SPI**).

Друга група пінів це **6 аналогових пінів**, які з'єднані з аналого-цифровим перетворювачем, вбудованим у плату Arduino**.**

Третя група пінів це піни живлення від зовнішніх джерел.

Контролер містить кварцовий резонатор на 16 МГц. Кнопкою скидання RESET можна перезапустити виконання програми. Світлодіод **L** це світлодіод налагодження (Debug), який дозволяє без додаткових компонентів перевірити працездатність контролера.

Для забезпечення інформаційного сполучення високо швидкісної роботи мікроконтролера з низько швидкісної роботою пристроїв периферії в контролерах використовують наступні поширені стандарти передачі даних.

I²C (**I I C**, від *англ.* **I**nter **I**ntegrated **C**ircuit) — стандарт передачі даних у вигляді послідовної асиметричної шини для зв'язку між інтегральними мікросхемами всередині електронних пристроїв. Це ***дводротовий інтерфейс***, який використовує дві двонаправлені лінії зв'язку (**SDA** і **SCL**) для внутрисхемного з'єднання периферійних компонентів з мікроконтролером.

SPI або **SPI bus** (**S**erial **P**eripheral **I**nterface— **С**ерійний **П**ериферійний **І**нтерфейс або **шина SPI**) це теж стандарт але для синхронної передачі даних. Поширена назва ***послідовний чотирьох провідний*** інтерфейс для 4х сигналів: **MOSI –** пін 11 (**M**aster-*передатчик* **O**ut **S**lave-*приймач* **I**n); **MISO –** пін 12 (**M**aster **I**n **S**lave **O**ut); **SCLK** (**S**erial **CL**oc**K**) – пін 13; **CS** – пін 10 (**C**hip **S**elect). **CS** — сигнал початку/завершення сеансу зв'язку (вибору веденого пристрою для передачі/читання даних. По завершенні обміну даних **CS** має бути знятий, що дозволить приймачу даних вийти з режиму читання/запису та перейти до режиму обробки даних. За допомогою наведених пінів здійснюється зв'язок SPI, для чого використовується бібліотека SPI.

Програми (скетчі) контролера Arduino UNO можуть бути завантажені через піни інтерфейсу **ICSP** (рис. 4.15,а) за допомогою додаткового USB-ASP AVR-програматора (на схеме не зображений). Але важливою особливістю контролерів Arduino є те, що програмування мікроконтролера, в тому числі і мікроконтролера **ATmega 328** виконується через **USB-порт**, без додаткового програматора.

Для завантаження скетчу (програми користувача) функцію програматора виконує PC і завантажувач Arduino. Завантажувач завантажує програму по універсальному асинхронному послідовному інтерфейсу **UART** (**U**niversal **A**synchronous **R**eceiver/**T**ransmitter) з використанням цифрових пінів першої групи **0** (**RX**) і **1** (**TX**). При цьому відбувається перетворення вхідних та вихідних байти даних в послідовний біт-потоки.

Завантажувач це фрагмент програмного коду, який записаний у зарезервовану ділянку пам'яті мікроконтролера на заводі-виробнику мікроконтролерів. Відразу після підключення плати Arduino запускається завантажувач і якщо в цей час завантажувач отримує команду програмування від **IDE** по інтерфейсу **UART**, то він завантажує програму у вільну ділянку пам'яті мікроконтролера. Якщо така команда не надходить, тоді запускається остання програма, яка знаходилася в пам'яті мікроконтролера Arduino.На платі Arduino UNO окрім мікроконтролера **ATmega 328** є додатковий мікроконтролер **ATmega 16U2** (рис. 4.15,а), який служить для сполучення або виконання функції інтерфейсу між USB-кабелем і контактами послідовного інтерфейсу UART і тому має ще назву мікроконтролер USB.

При підключення кабелем USB-B роз'єму плати Arduino UNO до USB-A роз'єму PC (на рис. 4.15,а не зображений) загоряється LED-світлодіод **ON** і починає блимати LED-світлодіод **L**, який підключений до піну 13 плати Arduino UNO.

При наявності контролера, наприклад, Arduino UNO**,** та його підключення USB-кабелем до комп'ютера і завантаження **Arduino IDE** (**IDE** – **I**ntegrated **D**rive **E**nvironment – інтегроване середовище розробки) доцільно приступати до індивідуального вивчення роботи зі скетчами Arduino у наступної послідовності:

1.В меню «**Інструменти**» спочатку відкрити і активізувати вкладку «**Плата**» → «**Arduino UNO**» (рис. 4.16), а потім обрати і активізувати в цьому меню вкладку «**Порт**» - «**COM3**», до якого підключена або треба

підключити плату Arduino UNO. В менеджері плат наведені існуючі типу контролерів з ключевим словом «**Arduino».**

2.Відкрити меню «**Файл**» → «**Приклади**» (рис. 4.17) і активізувати вкладку потрібного скетчу, наприклад, вкладку підключення скетчу **Stepper** (кроковий двигун) – **stepper oneRevolution** (програма – «Один оборот крокового двигуна»).

3. Відкрити меню «**Скетч**» → «**Підключити бібліотеку**» і активізувати вкладку бібліотеки, наприклад, для програми **Stepper** (рис. 4.18).

Бібліотека це набір функцій (кодів), які спрощують роботу з цільовим скетчем і які містять команди взаємодії із зовнішніми пристроями (модулями драйверів, двигунами, датчиками, LCD-екранами тощо). Наприклад, вбудована бібліотека **Stepper** дозволяє легко взаємодіяти з кроковими двигунами, а бібліотека **Servo** – із серводвигунами. Існує багато додаткових бібліотек, які можна завантажити з Інтернету і потім їх встановити

4. Перевірити працездатність плати при першому включенні можна за наступним алгоритмом:

4.1. **Обрати в меню «Файл» → «Приклади» → «01.Basics»** (рис. 4.16) скетч «**Blink**» («Блимати»):

4.2.Якщо у наведеному скетчі зменшити першу паузу при вмиканні LED-світлодіоду, наприклад, **`delay(500),`** а другу паузу збільшити при вимиканні LED-світлодіоду, наприклад, **`delay(3000)`** і ще раз компілювати програму, тоді LED-світлодіод повинен блимати з різними паузами при високому і низькому рівнях сигналу **`HIGH:=`**«**1**» і **`LOW:=`** «**0**».

4.3. Якщо в нижній червоної стрічки екрану після компіляції скетчу є свідчення правильної роботи завантажувача і тоді виконується результат по п.4.2 і можна рухатися далі для реалізації проекту.

Більшість плат Arduino мають бортовий світлодіод, який прикріплений до цифрового **піну** (шпильки) **13** . Після компіляції програма «**Blink**» виконує цикл з однаковими паузами вмикання і вимикання світлодіоду LED.

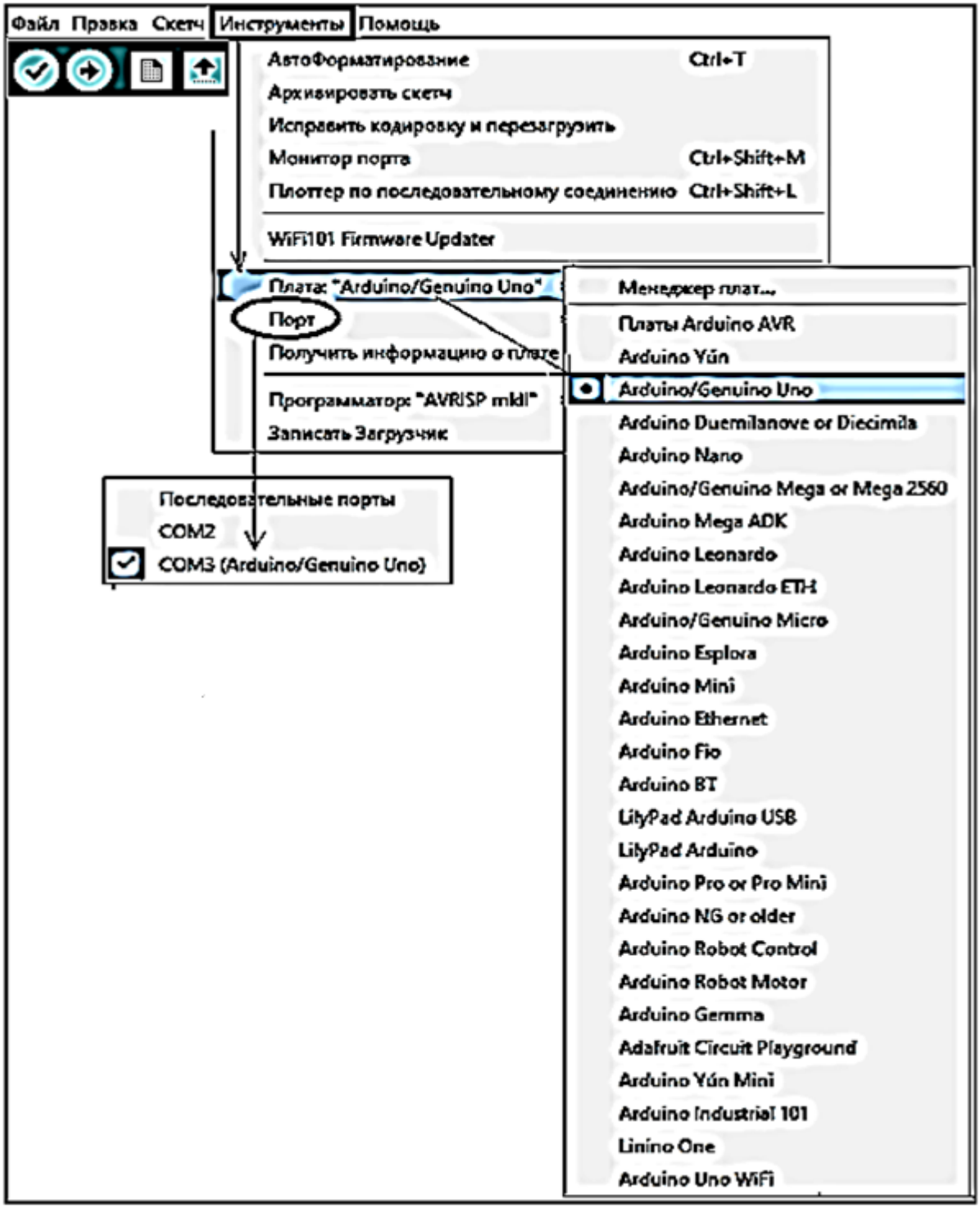
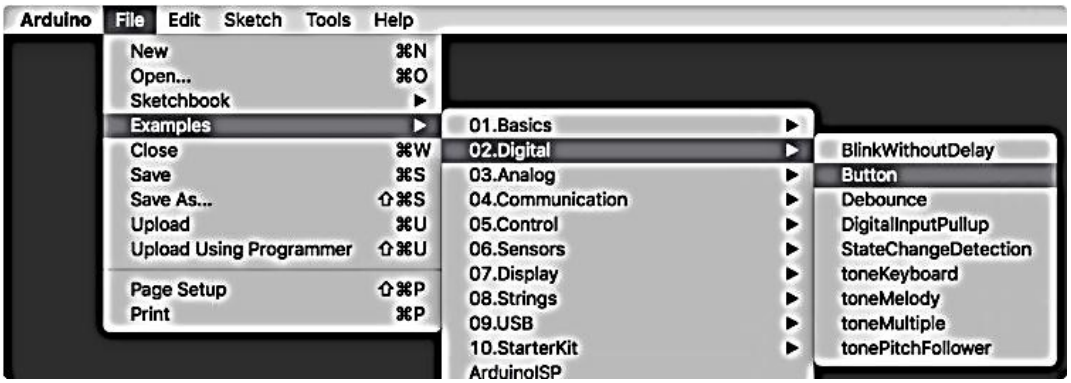
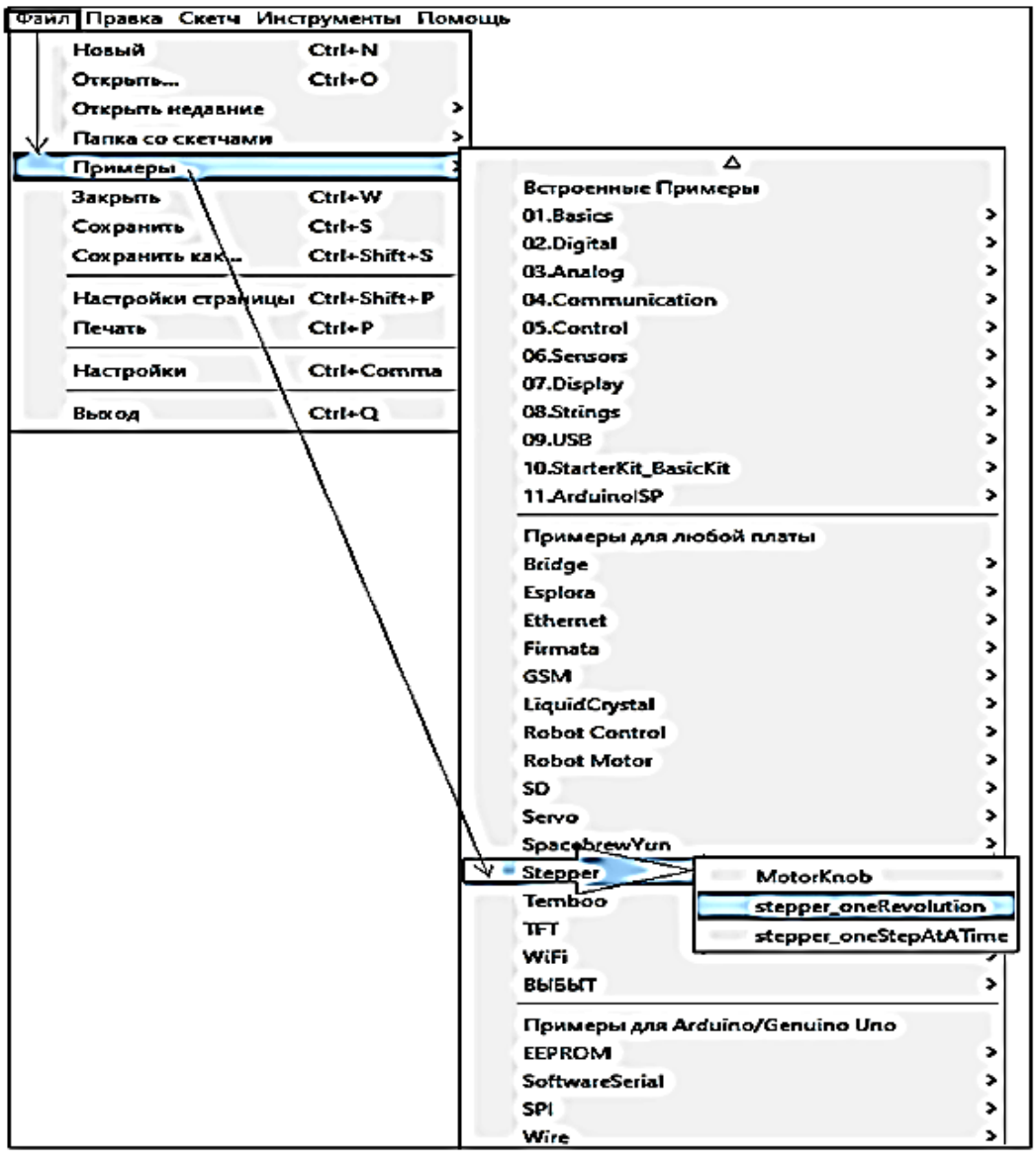


Рис. 4.16. Приклад підключення плати **Arduino UNO** і її підключення до потрібного **COM-порту** і менеджер плат контролерів родини Arduino



а



б

Рис. 4.17. Приклад підключення скетчів: а – **Button** (кнопка); б – **Stepper** (крокoвий двигун) – **stepper oneRevolution** (один оборот крокового двигуна)

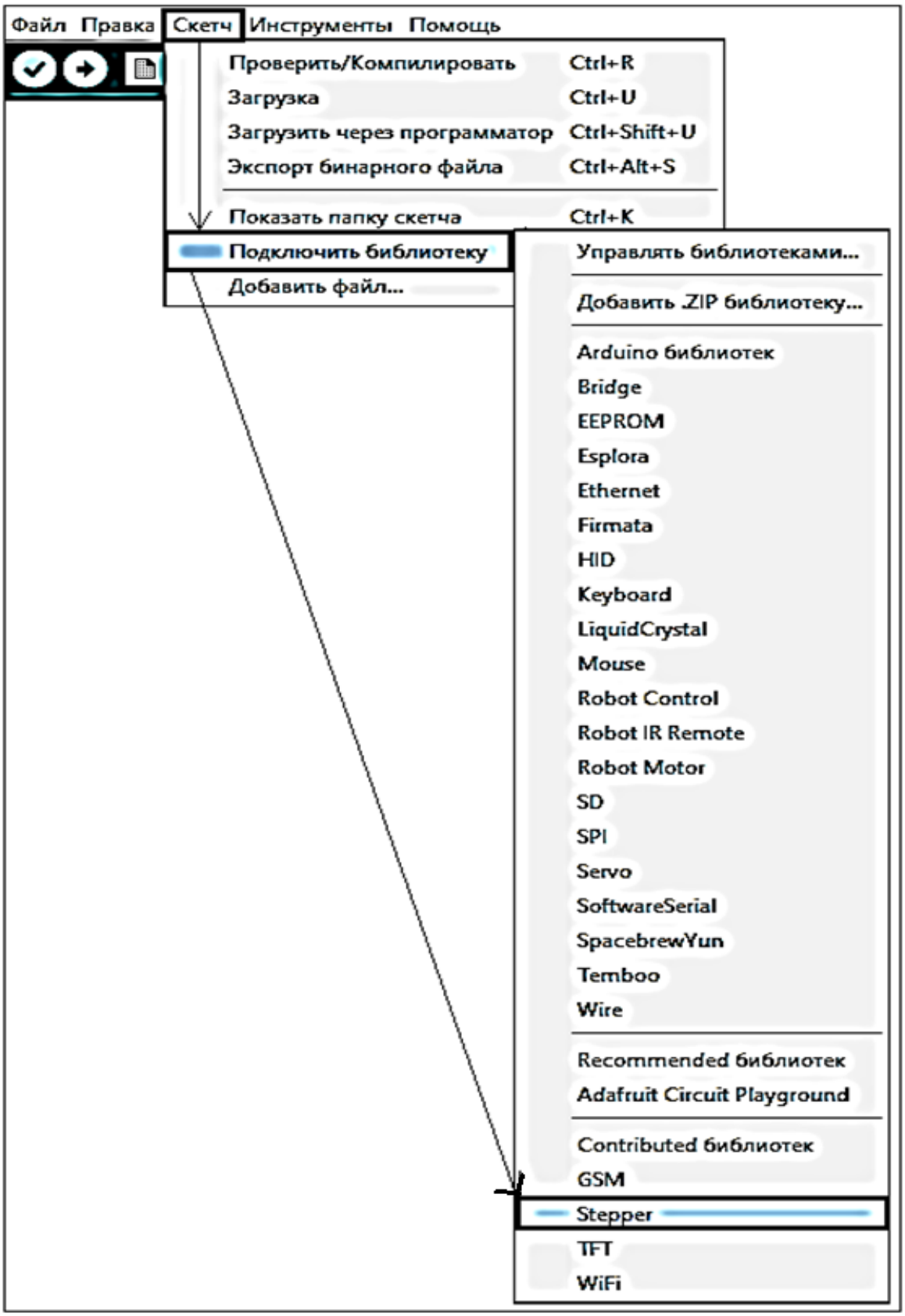


Рис. 4.18. Приклад підключення бібліотеки **Stepper**

Строкі скетчів (програм) мають вигляд з наступними коментарями двох виглядів /***...***/ або // .

```
    void setup() {
```

/*** **void setup ()** це функція налаштування без аргументів і тому має назву **void** (порожня), яка запускається один раз, коли натискається кнопка скидання RESET на платі та при включенні живлення плати ***/

```
   pinMode(LED BUILTIN, OUTPUT);
}
```

/*** **pinMode** це функція (аргументи в дужках), яка ініціалізує цифровий PIN-код 13 для вбудованого в плату світлодіоду, а саме **LED BUILTIN** (СВІТЛОДІОД ВБУДОВАНИЙ), як вихідний сигнал (**OUTPUT**) ***/

```
    void loop(){
```

/*** **void loop()** це функція циклу без аргументів і тому має назву **void** (порожня), яка запускається багато разів ***/

```
    digitalWrite(LED BUILTIN, HIGH);
```

/*** функція всередині циклу **loop** – увімкнути СВІТЛОДІОД ВБУДОВАНИЙ, даючи йому високий рівень напруги **HIGH** 5V ***/

```
    delay(1000);      // функція паузи – аргумент 1000 мс = 1 секунда
    digitalWrite(LED_BUILTIN, LOW);
```

/*** функція всередині циклу **loop** – вимкнути СВІТЛОДІОД ВБУДОВАНИЙ, даючи йому низький рівень напруги **LOW** 0V ***/

```
    delay(1000);      // функція паузи – аргумент 1000 мс = 1 секунда
}
```

4.4. **Сполучення і програмне керування кроковим приводом на платформі Arduino**

Широке застосування в мехатрониці і в робототехніці отримали драйвери на мікросхемах **ULN2003**, **L293D** та **L298N**. Драйвер **L293D** виконаний у вигляді здвоєного шилду і призначений для керування двома сервоприводами **DC** з редуктором та двома 5 вольтовими кроковими двигунами.

Шилди – це плати (модулі) розширення, які стикується з платою Arduino, як бутерброд. Функціонально шилди працюють за допомогою

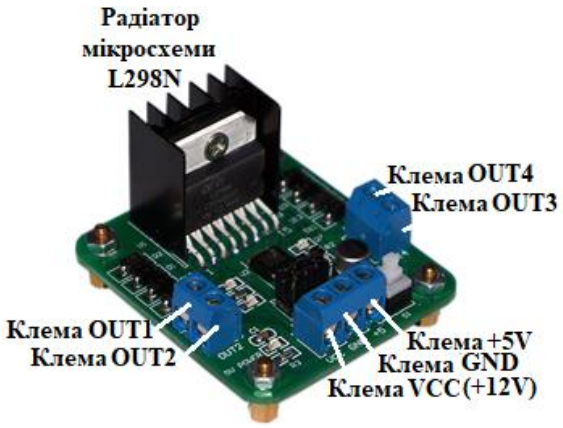
бібліотек. Бібліотеки призначені для поліпшення зв'язку і управління функціями шилдів і вони можуть організовувати інтерфейс з різними зовнішніми пристроями. Бібліотека це збірник підпрограм цільового призначення.

Драйвер **ULN2003** (рис. 19,а) призначений для підключення до контролера **Arduino UNO** одного крокового двигуна.

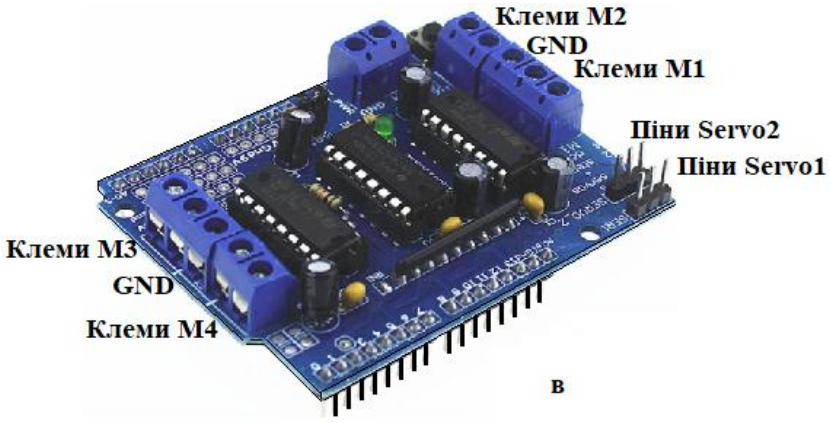
Драйвер **L298N** (рис. 19,б) призначений для підключення двох серводвигунів або одного крокового двигуна.

Драйвер **L298N** (рис. 19,в) призначений для керувати двома двигунами постійного струму і забезпечує максимальне навантаження до 2А на кожен двигун, а якщо задіяти паралельне включення двох драйверів для одного двигуна, то можна підняти максимальний струм до 4А.

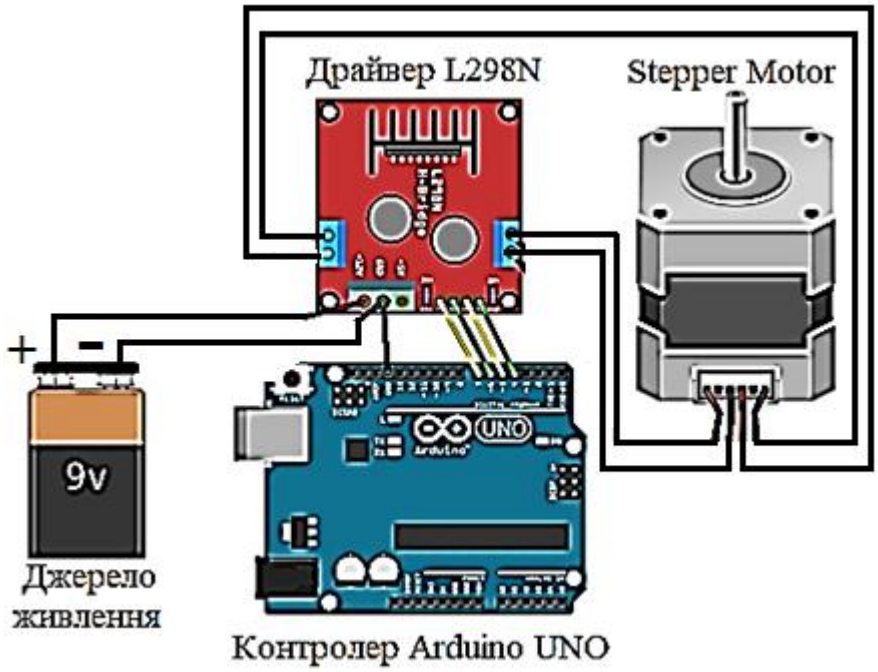
а



б



в



г

Рис. 4.19. Загальний вигляд поширених мехатронних модулів драйверів: а – драйвер **ULN2003** для підключення одного крокового двигуна; б – драйвер з радіатором охолодження мікросхеми **L298N** для підключення двох серводвигунів або одного крокового двигуна; в – шилд драйвера **L293D** для підключення двох серводвигунів **Servo1** і **Servo2**, двох крокових двигунів або чотирьох двигунів **M1…M4** постійного струму; г – приклад підключення крокового двигуна за допомогою драйвера **L298N** до плати **Arduino UNO** і зовнішньому джерелу живлення 9V

Приклад підключення до пінів плати **Arduino UNO** пінів та клем роз'єму модуля драйвера **ULN2003** та крокового двигуна (**Step Motor**) наведений в таблиці 4.1.

Можливі варіанти підключення живлення до контролера **Arduino UNO** наведені на рис. 4.20 .

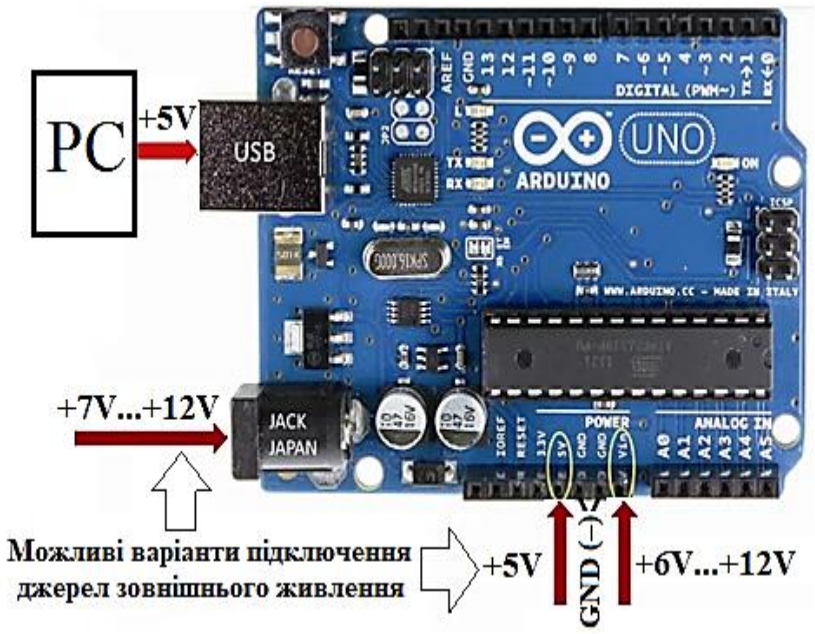


Рис. 4.20. Можливі варіанти підключення живлення до внутрішньої логіки контролера **Arduino UNO** від PC та до драйверів цільового призначення від зовнішніх джерел живлення постійної напругі

Приклад підключення до пінів плати **Arduino UNO** пінів та клем роз'єму модуля драйвера **ULN2003** та крокового двигуна (**Step Motor**) наведений в таблиці 4.1.

Таблиця 4.1. Сполучення пінів до рис. 4.19 і рис. 4.20

Модулі	Дроти						Роз'єм Stepper **28BYJ-48**				
	1 Си-ний (С)	2 Рожe-вий (Р)	3 Жов-тий (Ж)	4 Оран-жовий О	Чор-ний	Черво ний (Ч)	1 С	2 Р	3 Ж	4 О	5 Ч
Arduino UNO	**11**	**10**	**9**	**8**	**GND**	**Vin**	-	-	-	-	-
Драйвера **ULN2003**	**IN1**	**IN2**	**IN3**	**IN4**	**«-»**	**+5…12V**	**A**	**B**	**C**	**D**	**E**
Step Motor	-	-	-	-	-	-	**1**	**2**	**3**	**4**	**5**

Приклад сполучення крокового двигуна з драйвером **ULN2003**, контролером **Arduino UNO** та з персональним комп'ютером **PC** наведений на рис. 4.21.

В програмах на мові **Arduino-C** (рис.4.22 і рис.4.23) для навпіл крокового режиму комутації уніполярного крокового двигуна **28BYJ-48** у функції циклу **void loop ()** для відповідних виходів «**OUTPUT**», які підключені до виводів (піни) **2**, **3**, **4** та **5** контролера позначенні наступні рівні сигналів керування:

LOW (лёй) низький рівень **0V:=** «**0**»;
HIGH (хай) високий рівень **+5V:= «1».**

Кроковий двигун **28BYJ-48** постійного струму має 4 обмотки на статорі і 8 постійних магнітів на роторі, редуктор і широко використовується в робототехніці, **DIY**-пристроях (від *англ.* **D**o **I**t **Y**ourself - *зроби це сам*).

Основні технічні параметри крокового двигуна **28BYJ-48**:
• номінальна напруга живлення: 5 Вольт (постійний струм)
• кількість фаз: 4
• кількість зубців (постійних магнітів на роторі): 8
• кількість кроків: $n_{1.0} = 2 \cdot 4 \cdot 8 = 64$
• кількість мікро кроків у пів крковому режимі: $n_{0.5} = 64 \cdot 64 = 4096$
• крок: $\alpha = \frac{360}{64} = 5.625$ градусів
• номінальна частота: 100 Гц
• номінальний опір обмоток (при 25 Градусах): 50 Ом
• частота холостого ходу (за годинниковою стрілкою): 600 Гц
• частота холостого ходу (проти годинникової стрілки): 1000 Гц

Драйвер крокового двигуна дозволяє працювати як з 5V двигунами так і з 12V двигунами. На рис. 4.22 наведений **sketch_step 02** навпіл крокового режиму комутації уніполярного крокового двигуна з використанням бібліотеки **#incluide <Stepper.h> Arduino,** а на рис. 4.23 – без використання бібліотеки.

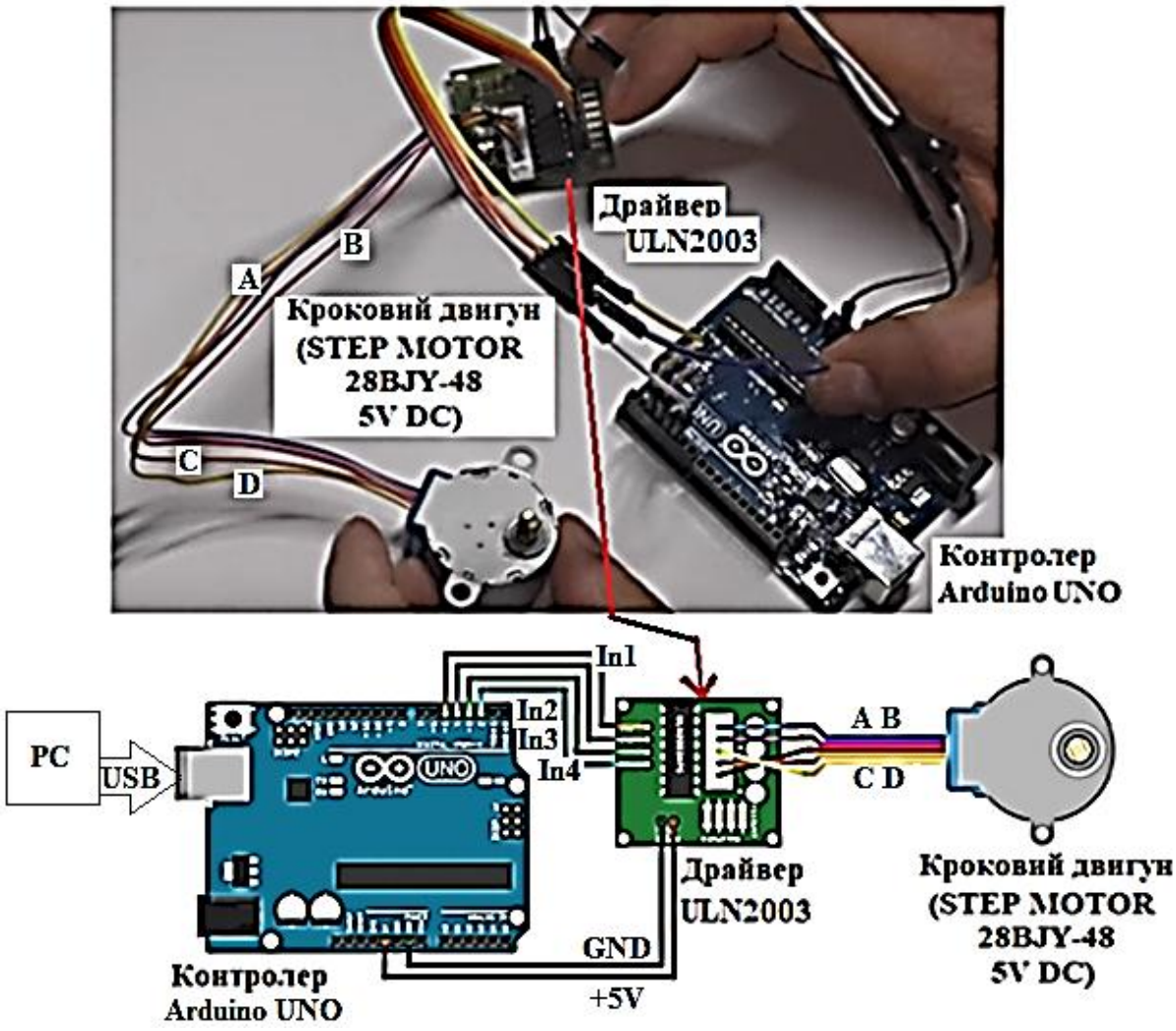


Рис. 4.21. Схема сполучення крокового двигуна **28BYJ-48** з драйвером **ULN2003** і контролером **Arduino UNO**

Тут і далі в програмах наведені прийняти наступні основні умовні позначення мови Arduino C :

sketch – скетч це поширена назва програм;

#incluuide <Stepper.h> – це директива «застосувати (включити)» бібліотеку <кроковий двигун> ;

// одно стрічковий коментар або **/***** багато стрічкові коментарі *****/** (на рис. 4.22 коментарі ні наведені);

{ … } – фігурні скобки виділяють фрагмент коду для компілятора. Блок коду завжди відкривається символом **{** і завершується символом **}**. Між фігурних скобок розташовуються команди ;

; – крапка з комою завершує команду. Якщо такий символ пропущений, компілятор Arduino видає повідомлення про помилку в програмі ;

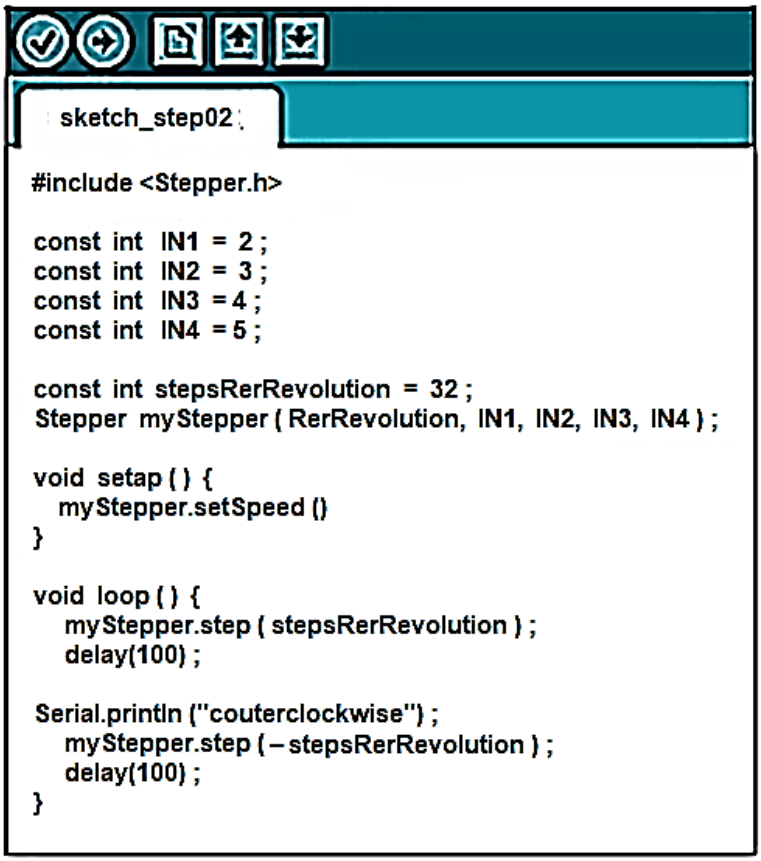


Рис. 4.22. Sketch_step02 навпіл крокового режиму комутації уніполярного крокового двигуна з використанням бібліотеки **#incluide <Stepper.h> Arduino (`IN1 = 2, IN2 = 3, IN3 = 4, IN4 = 5` –** номери пінів плати контролера Arduino, до яких підключені відповідні входи (**IN**put) крокового двигуна) ;

ТИПИ НАЙБІЛЬШ ПОШИРЕНИХ ЗМІННИХ:

`int` – це змінна типу «**int**eger» ціле число, складається з двох байтів (16 біт) і може приймати значення від -32 768 до +32 768 $\left(\frac{2^{16}}{2} = \frac{65536}{2} = 32\,768\right)$, наприклад: **`int ledPin = 13;`** (це світлодіод LED, який приєднаний до цифрового виводу (піну) 13);

`byte` – це змінна типу «байт» = 8 біт, а тому така змінна може приймати значення від 0 до 255;

boolen – це булева (логічна) змінна, яка може приймати два стану: **true** (істина) або **false** (*рос.* ложь) і займає в пам'яті 1 байт. Значення **true** відповідає логічної одиниці, а **false** – це не логічний нуль, як часто припускають, а стан який відрізняється від логічної одиниці;

char – це символьна змінна. Один символ позначається в одинарних лапках (апострофах), якщо декілька символів – в подвійних лапках, наприклад: **char MeineName = "B";** (символ **B** отримує номер 66 з набору символів стандарту A S C I I число 66 - розмір один байт) ;

float – це змінна з плаваючою комою і потребує в пам'яті 4 байта (32 біта значення числа зі знаком).

Поширені функції, команди, директив та операторів:

void setup () – це функція коду одноразового налаштування, а саме введення оголошених вище в скетчі змінних та інших параметрів тільки ***один раз*** ;

void loop () – це функція основного коду, щоб запустити цикл роботи скетчу, наприклад, крокового двигуна ***кілька разів*** ;

myStepper.step (stepsRepRevolution) – це код команди для включення 32 кроків роботи крокового двигуна за стрілкою годинника (рис.4.20) ;

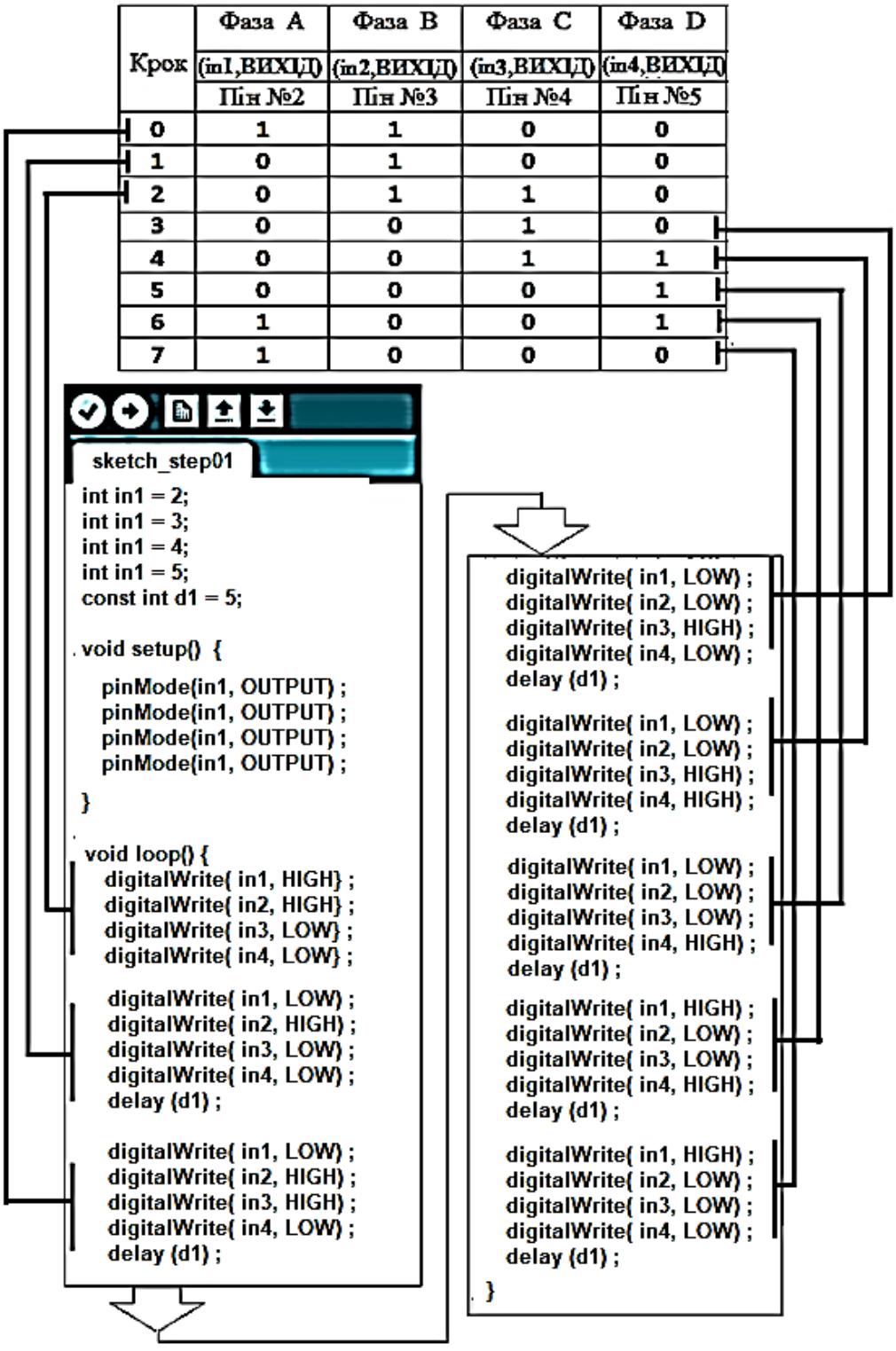
myStepper.step (–stepsRepRevolution) – це код команди для включення 32 кроків роботи крокового двигуна проти стрілки годинника або команда реверсу двигуна (рис.4.20) ;

delay (100) – це код команди переривання (затримки) на 100 мс (0,1 секунди) ;

#define (визначати) − директива препроцесора, яка запускається автоматично перед компіляцією наступного коду програми. Це ніби маленький компілятор, який заздалегідь перетворює команду #define в константи.

це знак **діез**. Перед командою він виконує функція як би маленького компілятор, який заздалегідь перетворює команду, наприклад **define** (визначити) в константи.

++ це поширений оператор для і н к р е м е н т а, а саме збільшення змінної «**i**» на одиницю, наприклад, **++ i** позначається збільшення кроків крокового двигуна на 1 при широтно-імпульсної модуляції (ШІМ) для регулювання швидкості крокового двигуна;



Крок	Фаза A (in1,ВИХІД) Пін №2	Фаза B (in2,ВИХІД) Пін №3	Фаза C (in3,ВИХІД) Пін №4	Фаза D (in4,ВИХІД) Пін №5
0	1	1	0	0
1	0	1	0	0
2	0	1	1	0
3	0	0	1	0
4	0	0	1	1
5	0	0	0	1
6	1	0	0	1
7	1	0	0	0

Рис. 4.23. Приклад таблиці логічних станів для **`sketch_step01`** навпіл крокового режиму комутації крокового двигуна з драйвером на рис. 4.21 без використання бібліотеки

-- це поширений оператор для д е к р е м е н т а, а саме зменшення змінної «**i**» на одиницю, наприклад, **- - i** позначається зменшення кроків крокового двигуна на 1 при ШІМ-регулюванні швидкості крокового двигуна.

== оператор дорівняння (дорівнюється, наприклад: **A == B**);

!= оператор не дорівняння (не дорівнюється, наприклад: **A != B**).

Повний перелік і призначення змінних, операторів, директив, циклів, функцій, команд і процедур можна знайти у мануалі до мови Arduino C, а деякі з них будуть далі пояснюватися в коментарях програм.

Вся символіка команд, операторів і операндів, функцій і процедур базується на таблиці кодів **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange - **А**мериканський **С**тандартний **К**од для **О**бміну **І**нформацією).

Для закріплення матеріалу наведений ще один приклад скетчу для навпіл керування кроковим двигуном **28BYJ-48** з коментарями, зміненими номерами штиркових контактів (пінів) і зміненою таблицею комутації обмоток:

/*** Оголошення змінної типу **integer** з ім’ям **motorPin1** для піну **12** контролера Arduino UNO означає, що що цей пін з’єднаний дротом синього кольору (**Pin1**) крокового двигуна ***/

```
int motorPin1 = 12;
```

/*** Оголошення змінної типу **integer** з ім’ям **motorPin2** для піну **11** контролера Arduino UNO з’єднаний з дротом рожевого кольору (**Pin2**) крокового двигуна ***/

```
int motorPin2 = 11
```

/*** Оголошення змінної типу **integer** з ім’ям **motorPin3** для піну **10** контролера Arduino UNO з’єднаний з дротом жовтого кольору (**Pin3**) крокового двигуна ***/

```
int motorPin3 = 10
```

/*** Оголошення змінної типу **integer** з ім'ям **motorPin4** для піну **9** контролера Arduino UNO з'єднаний з дротом помаранчевого кольору (**Pin4**) крокового двигуна ***/

```
int motorPin4 = 9;
```

/*** Оголошення змінної типу **integer** , що має ім'я **motorSpeed** для затримки **(delay)** часу **5** мс між комутаціями котушок, яка впливає на швидкість двигуна. При цьому дрот червоного кольору крокового двигуна з'єднаний з піном GND (ЗЕМЛЯ або **VCC** це 0V) контролера Arduino UNO ***/

```
int motorSpeed = 5;
void setup ()
{
```

/*** Команди **pinMode -** режиму одноразового оголошення виводів (пінів) крокового двигуна в якості виходів **OUTPUT** ***/

```
  pinMode (motorPin1, OUTPUT);
  pinMode (motorPin2, OUTPUT);
  pinMode (motorPin3, OUTPUT);
  pinMode (motorPin4, OUTPUT);
  Serial.begin (9600);
}
```

/*** надалі студентам надається можливість самостійно надати по строкові коментарі для кроків 1…8 при обертання крокового двигуна за стрілкою годинника ***/

```
void loop ()
{
      // крок 1 (код 1-0-0-0)
  digitalWrite (motorPin4, HIGH);
  digitalWrite (motorPin3, LOW);
  digitalWrite (motorPin2, LOW);
  digitalWrite (motorPin1, LOW);
    delay(motorSpeed);
      // крок 2 (код 1-1-0-0)
  digitalWrite (motorPin4, HIGH);
  digitalWrite (motorPin3, HIGH);
  digitalWrite (motorPin2, LOW);
  digitalWrite (motorPin1, LOW);
    delay(motorSpeed);
```

```
      // крок 3 (код 0-1-0-0)
  digitalWrite (motorPin4, LOW);
  digitalWrite (motorPin3, HIGH);
  digitalWrite (motorPin2, LOW);
  digitalWrite (motorPin1, LOW);
     delay(motorSpeed);
       // крок 4 (код 0-1-1-0)
  digitalWrite (motorPin4, LOW);
  digitalWrite (motorPin3, HIGH);
  digitalWrite (motorPin2, HIGH);
  digitalWrite (motorPin1, LOW);
     delay(motorSpeed);
       // крок 5 (код 0-0-1-0)
  digitalWrite (motorPin4, LOW);
  digitalWrite (motorPin3, LOW);
  digitalWrite (motorPin2, HIGH);
  digitalWrite (motorPin1, LOW);
     delay(motorSpeed);
       // крок 6 (код 0-0-1-1)
  digitalWrite (motorPin4, LOW);
  digitalWrite (motorPin3, LOW);
  digitalWrite (motorPin2, HIGH);
  digitalWrite (motorPin1, HIGH);
     delay (motorSpeed);
       // крок 7 (код 0-0-0-1)
  digitalWrite (motorPin4, LOW);
  digitalWrite (motorPin3, LOW);
  digitalWrite (motorPin2, LOW);
  digitalWrite (motorPin1, HIGH);
     delay (motorSpeed);
       // крок 8 (код 1-0-0-1)
  digitalWrite (motorPin4, HIGH);
  digitalWrite (motorPin3, LOW);
  digitalWrite (motorPin2, LOW);
  digitalWrite (motorPin1, HIGH);
     delay(motorSpeed);
}
```

4.5. **Сполучення і програмне керування сервоприводами та двигунами постійного струму з використанням ШІМ і контролера Arduino UNO**

В деяких конструкціях мехатроніки і робототехнічних систем потрібно повертати ведену ланку на певний запрограмований кут повороту. Для цієї мети також використовується один або два сервопривода із широтно-імпульсною модуляцією, приклад якої наведений на рис.4.24. При цьому ширина імпульсу змінюється від 1 мс до 2 мс через кожні 20 мс (50 Гц). Тобто при періоді **T = 20 мс = const** часова ширина імпульсу $t_{\text{імп}}$ і ширина паузи $t_{\text{паузи}}$ змінюються але завжди зберігається наступна умова: $\boldsymbol{T = t_{\text{імп}} + t_{\text{паузи}} = const}$.

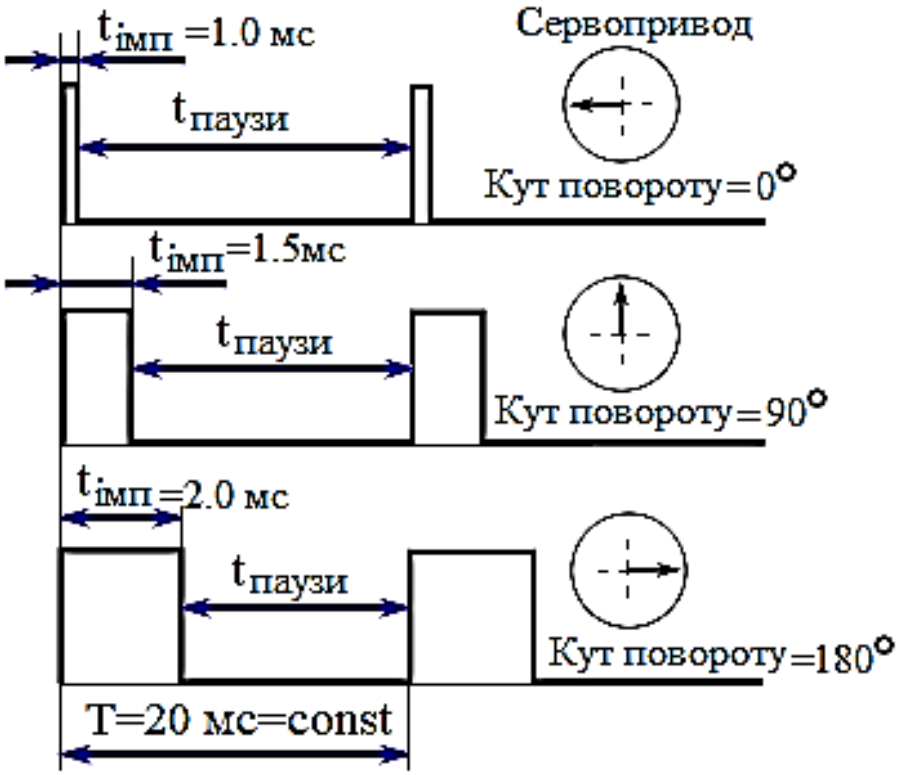


Рис. 4.24. Форма сигналів **ШІМ** (**PWM**) керування сервоприводом зі зворотним зв’язком на платформі Arduino UNO

Положення ротора сервоприводу вимірюється внутрішнім потенціометром пристрою і повертається в плату управління контролера Arduino, який програмно коригує положення ротора в залежності від отриманого значення сигналу.

Як було підкреслено на початку розділу сервопривод – це привод з керуванням через негативний зворотний зв'язок, що дозволяє точно керувати параметрами руху. Сервопривод мають три контактні виводи (дроти): живлення (червоного кольору), GND (земля) – коричневого або чорного кольору і сигнальний вхід – дрот білого або оранжевого кольору.

Керування сервоприводами здійснюється по сигнальної лінії за допомогою прямокутних імпульсів тривалість яких можна регулювати. У сервоприводах подача імпульсу тривалістю 1 мс призводить до установки сервоприводу в положення 0, імпульс тривалістю 1,25 мс встановлює сервопривод в положення 45 град, 1.5 мс - 90 град, 2 мс - 180 град. Після того як імпульс поданий, вал сервоприводу встановлюється в певній позиції і залишається там до надходження наступної команди.

Тестова програма керування швидкістю і напрямом обертання двох серводвигунів (рис.4.25,а) має наступний вигляд:

```
#include<Servo.h> //директива застосування бібліотек серводвигунів

Servo myServo1;   //Створити об'єкт myServo1 для сервопривода 1
Servo myServo2;   // Створити об'єкт myServo2 для сервопривода 2
int potpin1 = 0;          //Аналоговий пін №0 для потенціометра 1
int potpin2 = 1;          //Аналоговий пін №1 для потенціометра 2
int val1;                 //Змінна 1 для читання значення з піну №0
int val2;                 //Змінна 2 для читання значення з піну №2
void setup() {
  myServo1.attach(9); //Підключення Servo1 на цифровий pin №9
  myServo2.attach(10);//Підключення Servo2 на цифровий pin №10
}
void loop() {
  val1 = analogRead(potpin1);
                    //Зчитування значень від 0 до 1023 потенціометра 1
  val1 = map(val1, 0, 1023, 0, 179);
                    //Масштабування потенціометра 1, щоб
                    //використовувати значення від 0 до 179 для Servo1
  myServo1.write(val1);
                  //Встановлення Servo1 відповідно до значень масштабу
  delay(15);   //Затримка 15 мс
```

```
 val2 = analogRead(potpin2);
                    //Зчитування значень від 0 до 1023 потенціометра 2
 val2 = map(val2, 0, 1023, 0, 179);
                     //Масштабування  потенціометра 2, щоб
                     //використовувати значення від 0 до 179 до Servo2
  myServo2.write(val2);
                  //Встановлення Servo1 відповідно до значень масштабу
 delay(15);      //Затримка 15 мс
}
```

Зчитування значень від **0** до **1023** потенціометрів програмно підключаємих до аналогових пінів вбудованого (внутрішнього) АЦП (**ADC - A**nalog **D**igital **C**onverter) відбувається наступним чином. Аналого-цифровий перетворювач 10-розрядний ($2^{10} = 1024$). І тому його крок зміни U_{step} при опорної напрузі 5V дорівнюється значенню аналогової напрузі на виході контролера Arduino, яке дорівнюється **5V/1024=0.00488V=4.88 mV**. При зчитуванні значень з аналогових пінів потенціометрів від **0** до **179** напруга на обмотках серводвигунів змінюється від **0V до 4.88mV*179 = 873.52 mV = 0.87 V**, що відповідає певному куту повороту ротора **Servo1** і **Servo2**.

Функції **`analogRead(potpin1)`** і **`analogRead(potpin2)`** зчитують значення аналогових входів потенціометрів з розрішенням 10 біт і встановлює змінну **`value`** рівної 179 = 0.87 вольт. Аналогові піни 0…5, на відміну від цифрових 0…13, не потрібно оголошувати, як входи на початку програми.

Керування сервоприводами здійснюється за допомогою змінних резисторів. Різкість повороту, як і точність і кут повороту залежать від номіналів потенціометрів. Монтажна схеми з макетною (без паяльною) платою наведена на рис.4.25.

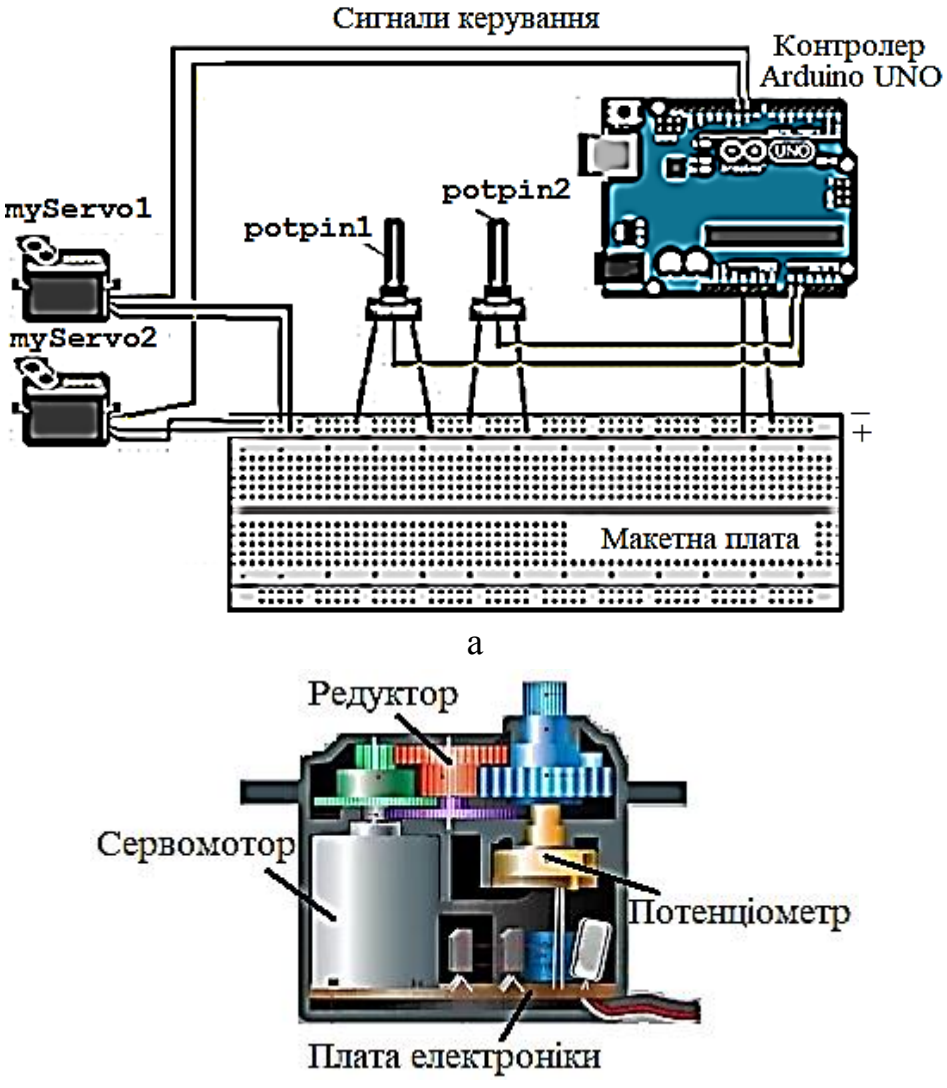
Деталі, необхідні для збирання: плата Arduino UNO, або Arduino Nano, або Arduino Mega, або інша; два сервоприводу **Micro Servo SG 90**; два потенціометра (змінні резистори); макетна плата (**Bread Board**), сполучні дроти-перемички «мати-тато».

Для установки двох сервоприводів SG90 зручна платформау вигляді 2D-кронштейну типу **2-Axis FPV** для програмованого керування

положеннями датчиків або камери, контроль над якими здійснюється за допомогою Arduino UNO та інших контролерів Arduino.

Монтажна схема непрямого керування з драйвером двома двигунами **D1** і **D2** постійного струму наведена на рис. 4.26, яка містить плату Arduino UNO і драйвер на мікросхемі **L298N**.

В схемі на рис. 4.26 наведені наступні позначення пінів: **VCC** – підключення зовнішнього живлення двигунів; **+5В** – напруга живлення; **GND** – земля (**G**rou**ND**); **IN1, IN2, IN3, IN4** – входи (**IN**put) керування двигунами; **OUT1, OUT2** – вихід (**OUT**put) першого двигуна D1; **OUT3, OUT4** – вихід другого (**OUT**put) двигуна D2;



а

б

Рис. 4.25. Монтажна схема (а) з макетною платою та двома сервоприводами **SG90** (б) з вбудованим редуктором і потенціометром зворотного зв’язку прямого ручного керування за допомогою

потенціометрів 1 і 2 (сервоприводи і потенціометри позначені як у наведеної)

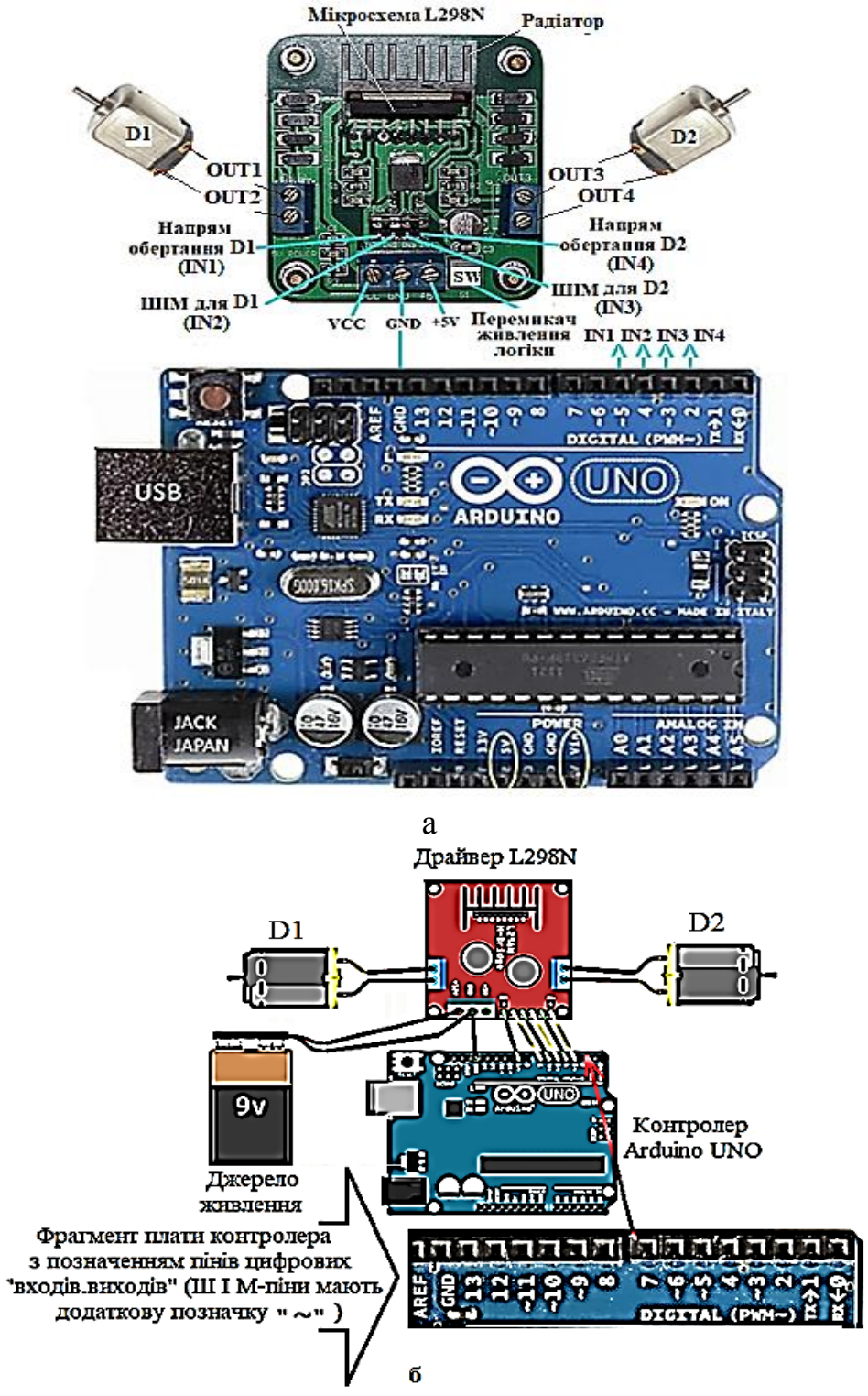


Рис. 4.26. Приклад монтажної схеми підключення до контролера Arduino UNO драйвера на мікросхемі **L298N** непрямого керування двома двигунами **D1** і **D2** постійного струму**: а** – без підключеного джерела живлення; **б** – з підключенним джерелом живлення і фрагментом плати контролера

Широтно-імпульсна модуляція (**ШІМ**) дозволяє плавно змінювати швидкість обертання двигуна. Для ШІМ керування на відповідний вхід потрібно подати логічну «1». На який саме вхід IN1 або IN2 подане сигнал ШІМ, або напрям обертання - різниці не має. Теж саме справедливо і для входів IN3, IN4.

SW – перемикач живлення логічної частини мікросхеми драйвера. Якщо SW включений – живлення логічної частини відбувається від внутрішнього джерела напруги. При вимкненому S1 живлення відбувається від зовнішнього джерела напруги.

Для живлення логічної частини схеми необхідно натиснути кнопку SW (рис.4.25,а) або вставити перемичку (залежить від типу модуля). Якщо ж на модулі не передбачений перетворювач електричної напругі для отримання +5M, то додатково необхідно з'єднати вихід +5V від плати Arduino до входу +5V драйвера.

Мікросхема L298N являє собою здвоєний мостовий драйвер двигунів і призначена для управління двома сервоприводами DC або одним кроковим двигуном. Ця мікросхема знаходить широке застосування в робототехніці. Одна мікросхема L298N забезпечує максимальне навантаження до 2A на кожен двигун, а якщо задіяти паралельне включення для одного двигуна, то можна підняти максимальний струм до 4A.

Широтно-імпульсна модуляція (ШІМ) дозволяє плавно змінювати швидкість обертання двигуна. Для ШІМ управління на відповідний вхід потрібно подати логічну «1». На який саме вхід IN1 або IN2 поданий сигнал ШІМ, або напрям обертання - різниці не має. Теж саме справедливо і для входів IN3, IN4.

Як випливає зі схеми на рис. 4.26,а з драйвером **L298N** для підключення двох двигунів D1 і D2 піни IN1 і IN4 драйвера використовуються для програмування напрямку руху за стрілкою годинника або проти, а піни IN2 і IN4 - для ШІМ-керування швидкістю серводвигунів.

Тестова програма керування швидкістю і напрямом обертання двох двигунів постійного струму (рис.4.26) має наступний вигляд:

```
#define D1 2              //визначити напрямок обертання двигуна D1
#define M1 3              //визначити ШІМ-керування двигуном D1
#define D2 4              //визначити напрямок обертання двигуна D2
#define M2 5              //визначити ШІМ-керування двигуном D2
// #define це відповідні директиви «застосувати (включити)» для
//призначення напрямку обертання та ШІМ-керування швидкістю
//відповідних двигунів

bool direction = 0;       //поточний напрямок обертання
int value;                //поточне значення ШІМ
void setup()              //виконати один раз при запуске
{
 pinMode(D1,OUTPUT);      //конфігурація порту для D1,  як ВИХІД
 pinMode(D2,OUTPUT);      //конфігурація порту для D2, як ВИХІД
}
void loop()               //виконати цикли для режимів роботи
{
 for(value = 0; value = 255; value+=1)
 digitalWrite(D1, direction);        //Напрям обертання D1

 digitalWrite(D2, direction);        //Напрям обертання D2

 analogWrite(M1, value);             //Швидкість обертання D1

 analogWrite(M2, value);             //Швидкість обертання D2

   delay(20);                        // Затримка 20 мс
 direction=direction^1; //Інвертуємо значення, щоб у наступному
                        //циклі обертатися в іншу сторону
 }
```

Сервопривод побудований на засадах крокового двигуна має назву «**інтегрований сервопривод**» тип СПК (сервопривод кроковий). Побудова структури такого сервоприводу наведена на рис. 4.27.

Інтегрований сервопривод побудований на базі гібридного крокового двигуна, в якому використовується без крокове (векторне) керування на

основі адаптованого спеціально для крокових приводів. Встановлення параметрів і робота СПШ відбувається під програмним забезпеченням «МотоМастер»

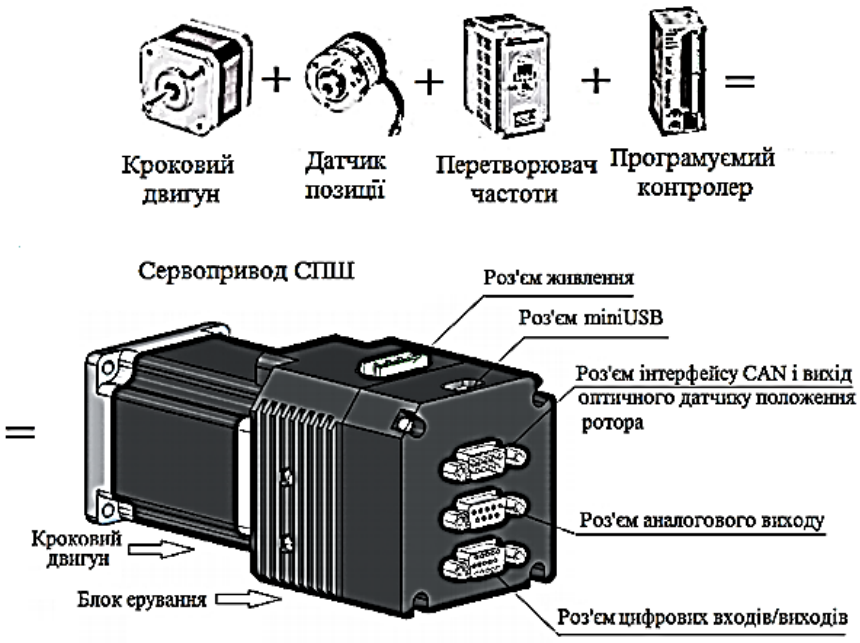


Рис. 4.27. Складові утворення структури інтегрованого сервоприводу

4.6. **Візуальне програмування FLProg для платформи Arduino**

Для сервісного обслуговування сучасних технологічних машин і верстатів з цифровими системами керування кроковими двигунами та сервоприводами майбутнім інженерам-механікам потрібен спрощений вхід в такі мехатронні системи. Навчання програмуванню доцільно починати з вивчення систем візуального програмування. Одної з поширених таких систем програмування є програма **FLProg** (рис. 4.26) для контролерів на платформі Arduino.

На рис. 4.28,а і рис. 4.28,б наведені застосування активних опцій: «*рос.* Защита от дребезга» (Захист від брязкоту) та «*рос.* Включить подтягивающий резистор» (Включити підтягаючий резистор), мета і зміст яких наступний.

З метою підвищення надійності роботи (перешкода стійкості) мікроконтролера в схемах мехатроніки з контролером застосовується підтягуючий резистор 1 кОм, призначення якого пояснюється схемами на рис.4.29. На рис. 4.29,а наведена схема підтягування резистором R1 до входу 1 мікросхеми логічного «0», а саме GND» (землі) при ненатиснутої

кнопки S1 (поширена англомовна назва «**pull down**» – тягнути вниз), а на рис. 4.29,б – резистор R2 підтягує на вхід 1 мікросхеми логічну «1» при ненатиснутої кнопки S1(поширена англомовна назва «**pull up**» – тягнути вгору). На схема рис. 4.29,в і рис. 4.29,г кнопка S1 умовно як би висить у повітрі і сигнал на вході 1 мікросхеми може бути невизначеним, наприклад, при появі випадкових зовнішніх електромагнітних збуджень і в результаті може відбутися хибне спрацьовування мікросхеми від сигналів між «0» і «1» при ненатиснутої кнопки S1.

На рис. 4.28 наведені фрагменти робочого поля проекту для програмування крокового двигуна в програмі **FLProg** на мові **FBD** для наступної компіляції програми на мові **Arduino_C.**

Програма **FLProg** працює на комп'ютері під керуванням операційних систем Windows, Linux-32 або Linux-64. При створенні нового проекту спочатку треба вибрати мову програмування (**FBD** або **LD**), на якій ви будете створювати проект, і контролер, на якому цей проект буде реалізований.

Дребезг контактів виникає через особливості конструкції кнопки. Всякий раз при її натисканні і відпусканні сигнал приймає низький або високий рівень не відразу, а поступово, що викликає коливання контактів, тобто дребезг контакту. Контролер працює настільки швидко, що брязкіт сприймається як кілька послідовних "0" і "1".

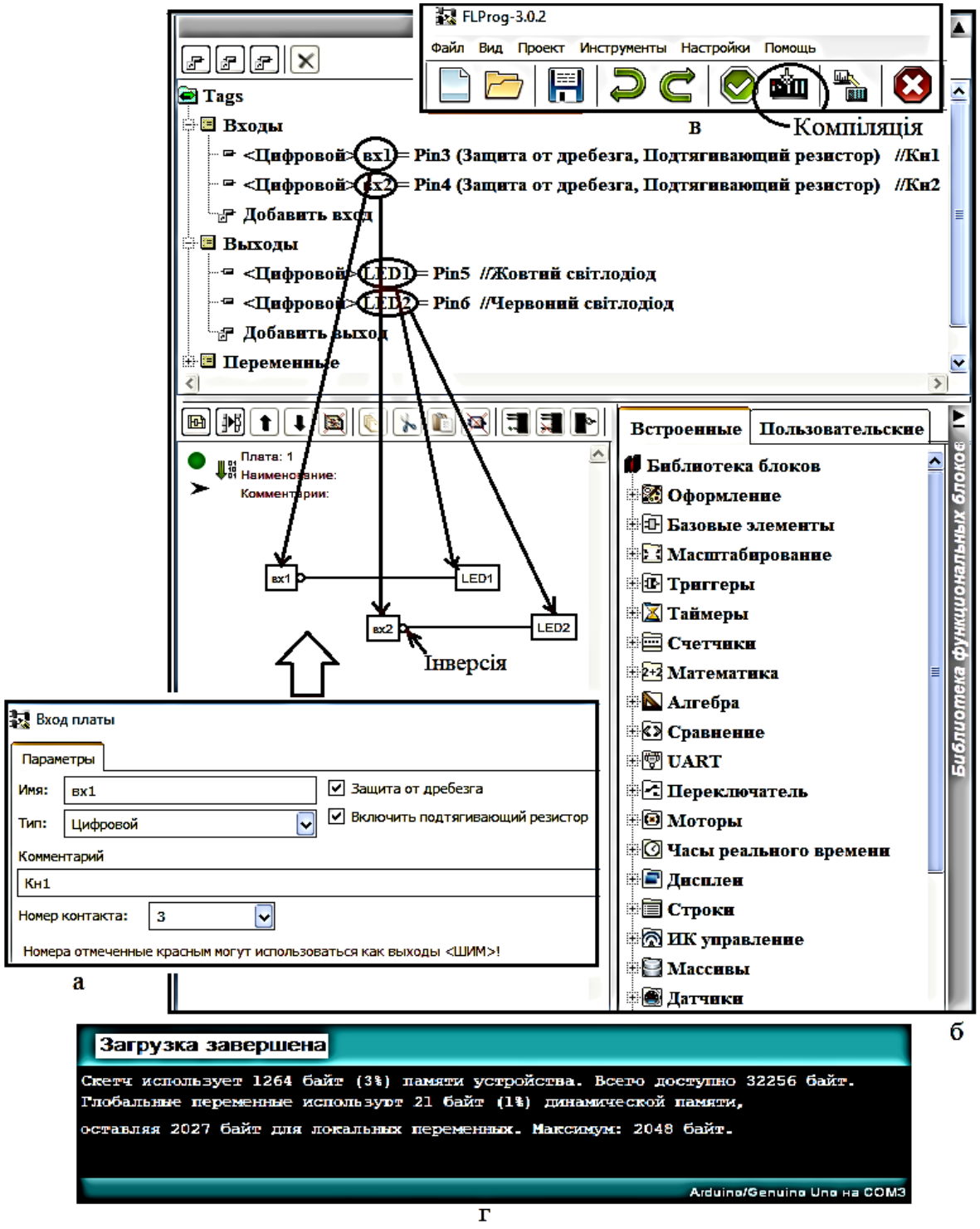


Рис. 4.28. Фрагменти візуального програмування комутації двох кнопок пуск з інверсією в програмі FLProg на мові FBD : а – введення змінних; б – перетягування змінних на робоче поле, які перетворюються у блоки(плати) і меню блоків (плат) – програмних компонентів програми FLProg; в – кнопка компіляції програми FLProg ; г - повідомлення про завершення вдалої компіляцію програми Arduino

При активізації опцій «Защита от дребезга» і «Включить подтягивающий резистор» (рис. 4.29) підтягуючий резистор і дребезг контактів усуваються програмно на фізичному рівні контролера Arduino.

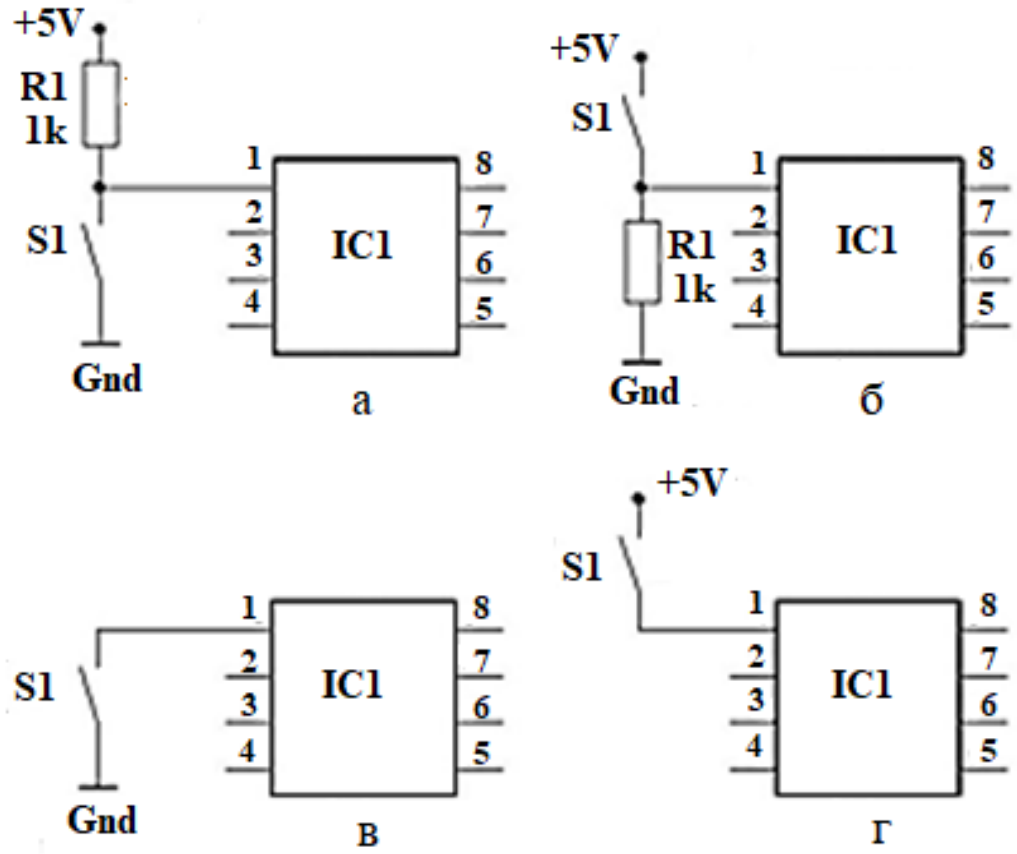


Рис. 4.29. Електричні схеми **з** підтягуючим резистором R1 (рис.а - **pull down** і рис.б - **pull up**) і без підтягуючого резистора (рис.в і рис.г)

Проект в **FLProg** являє собою набір плат (іконок), на кожній з яких зібраний закінчений модуль загальної схеми. Для зручності роботи кожна плата має найменування і коментарі. Фрагменти вигляду вікон програми FLProg при роботі в режимі мови FBD наведений на рис. 4.30, а приклад програми на мові **Arduino_C** після компіляції програми **FLProg** на мові **FBD** наведений на рис. 4.31.

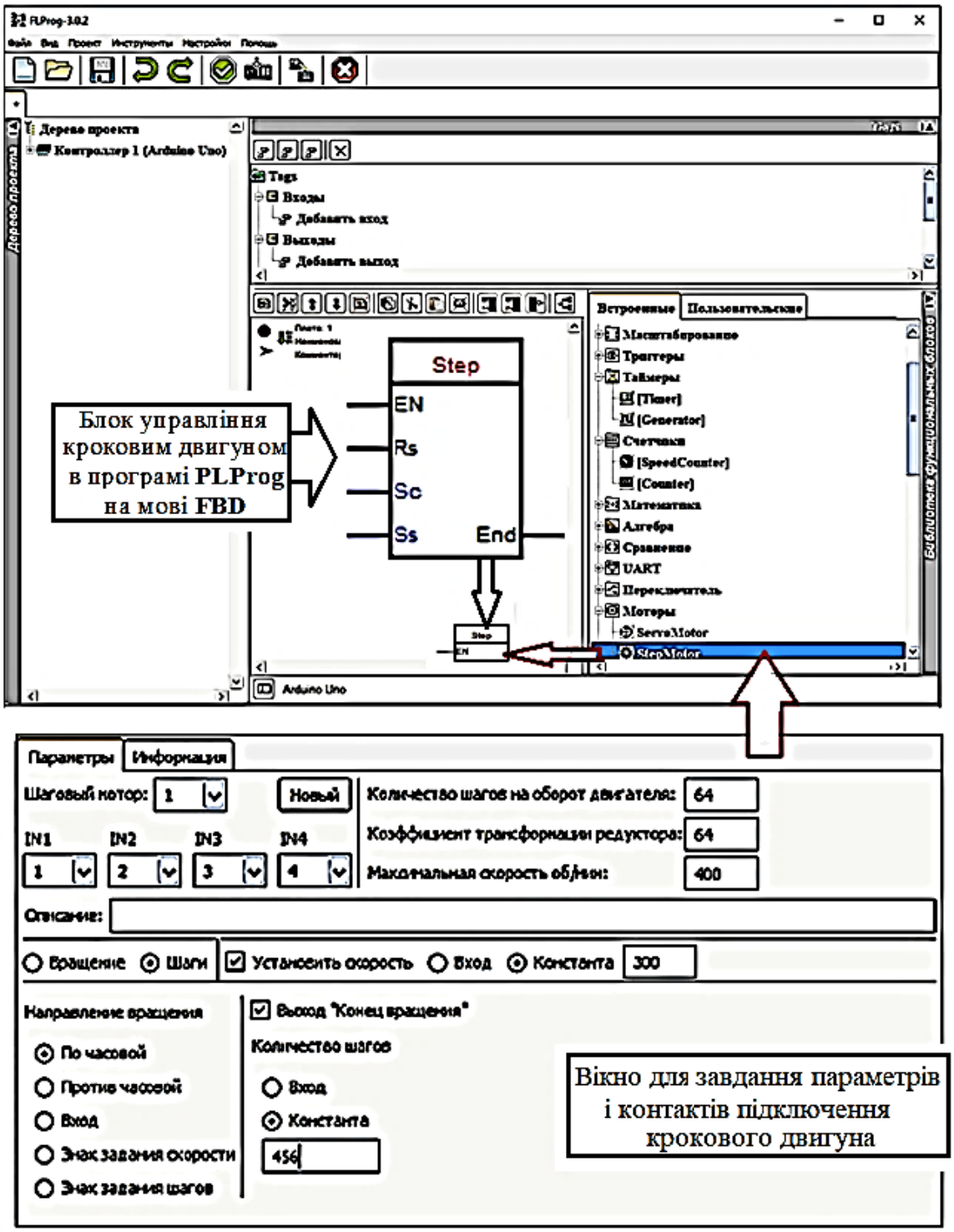


Рис. 4.30. Фрагменти робочого поля проекту для програмування крокового двигуна в програмі **FLProg** на мові **FBD** для наступної компіляції програми на мові **Arduino C**

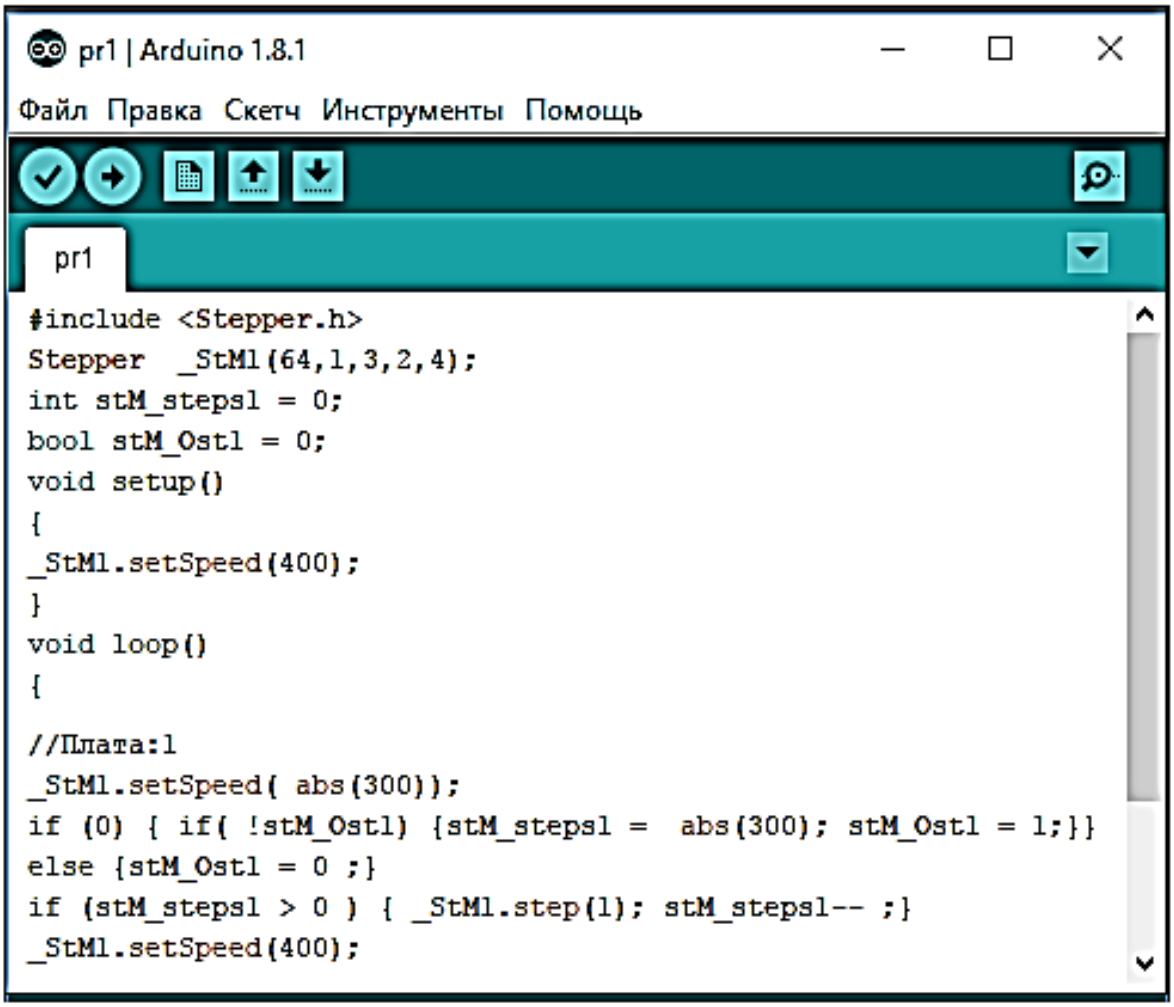


Рис. 4.31. Програма на мові **Arduino_C** після компіляції програми **FLProg** на мові **FBD**

Для отримання запрограмованих точних положень (кутів повороту) і швидкостей робочих інструментів у автоматизованих технологічних машинах і CNC-верстатках або зупинення веденої ланки у програмно керованих механізмах в мехатронних і робототехнічних системах з цифровими системами керування при роботі з платформами Arduino широко використовуються сервопривод з редуктором, наприклад, типу **Micro Servo SG90**. Два таких сервопривода можуть працювати з шилдом драйвера **L293D** (рис.4.19,в) на платі Arduino UNO.

Сервопривод (Servo) це регулюємий привод на засадах безколекторного двигуна постійного або змінного струми, який містить на валу ротора **датчик зворотного зв‘язку** кута повороту (кутової швидкості, моменту на валу в залежності від зусилля навантаження з боку машини), наприклад, у вигляді **енкодера** та має вбудований **контролер** для

автоматичного підтримання запрограмованої величини кута повороту ротора.

Micro Servo SG90 має вбудований міні-редуктор, два виводи (+5V і GND) для живлення напругою постійного струму і один – для завдання значення позиції (кута повороту). Обертання сервоприводу задається за допомогою імпульсів, які мають тривалість від 1 до 2х мс через кожні 20 мс . Досягнуте положення ротора сервоприводу вимірюється внутрішніх потенціометром пристрою і повертає це значення по зворотного зв'язку в плату управління. Зворотній зв'язок коректує положення ротора в залежності від отриманого значення сигналу.

4.7. Широтно-імпульсна модуляція (ШІМ) регулювання частоти обертання ротора двигунів мехатроніки

В технічної літературі поширена скорочена назва **ШІМ - Ш**иротна **І**мпульсна **М**одуляція (*англ.* **PWM – P**ulse **W**idth **M**odulation), яка програмно реалізується контролером Arduino UNO і використовується для отримання регульованої електричної напругі при регулюванні швидкості роботи технологічних машин з сервоприводом.

Важливим параметром ШІМ-регулювання швидкості крокового приводу і сервоприводу на засадах крокового двигуна є шпаруватість ШІМ-сигналів керування, яка задається програмно наступним чином:

Як змінюється форма сигналів керування швидкістю сервоприводу при різної шпаруватості ШІМ-сигналів керування наведене на рис. 4.32.

Шпаруватість (*рос*. скважность) **сигналу** (***S***) - це відношення періоду сигналу ***T*** до його тривалості ***t1***

$$S = \frac{T}{t1}$$

На виході отримується електрична напруга, яка еквівалентна площі під імпульсом. Для регулювання швидкості виконавчих механізмів потрібно **програмно змінювати значення ШІМ, які пов'язані із шпаруватості імпульсів.** При цьому змінюється еквівалентна електрична

напруга для підключених до виходів Ш І М контролера виконавчих механізмів обертової або зворотно-поступової дії.

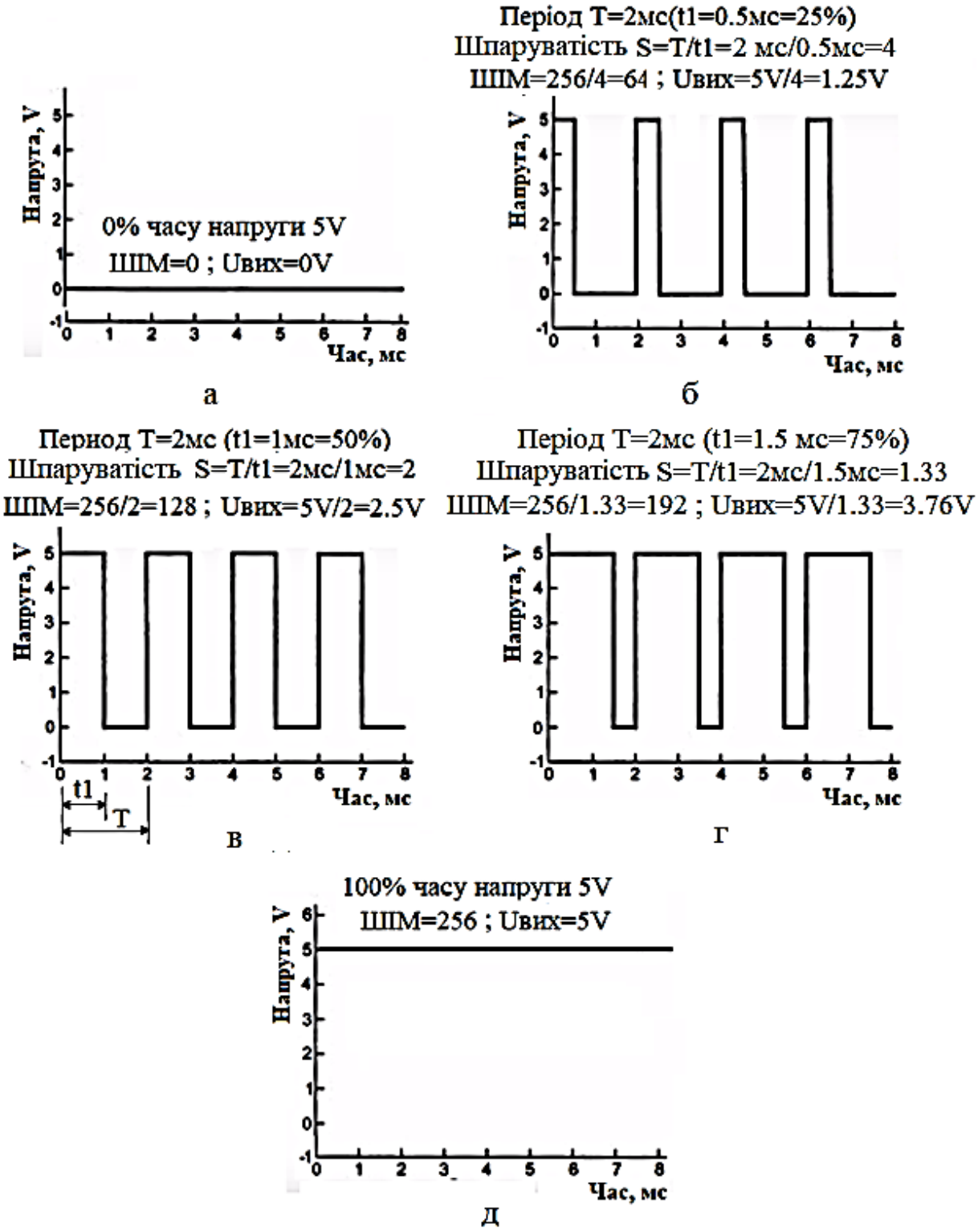


Рис. 4.32. Форма сигналів **ШІМ** (**PWM**) при різної шпаруватості імпульсів

Як випливає з рис. 4.32 при ШІМ частота сигналу керування залишається однаковою, а змінюється лише співвідношення тривалості сигналу високого рівня (**t1: U=5V**) та сигналу низького рівня (**t2=T-t1:= U=0V**).

Розглянемо і проаналізуємо спрощений скетч для ШІМ регулюванням яскравості світіння одного світлодіоду на платформі Arduino UNO. Світлодіод в поширеному позначенні **LED** (*англ.* **L**eith **E**missions **D**iode) це напівпровідниковий компонент, який створює оптичне випромінювання при пропусканні скрізь нього електричної струми в прямому напрямку. Підключення світлодіоду відбувається із врахуванням полярності (+5V анод – електрод більшої довжини ніж електрод катоду, «мінус» – катод) з використанням резистора. Пряма напруга для LED з діаметром лінзи 5 мм становить 2V і номінальний робочий струм 20 мА (0.02А). Вважаємо, що високий рівень напруги на виході плати Arduino UNO становить 5V. Щоб задати необхідний струм в 20 мА, обчислюємо величину опору *R*:

$$R = \frac{U_r}{I_r} = \frac{5V - 2V}{0.02A} = 150 \text{ Ом}$$

Значить, резистор повинен бути приблизно такого або більшого номінального опору. Для складання монтажної схеми потрібні: плата Arduino UNO з кабелем USB, макетна плата без паяльна, LED, резистор на 150…220 Ом, 2 перемички (мама-мама), перехідник крона-роз'єм живлення і зовнішнє джерело живлення +7V…+9V або батарейка Крона на 9V).

Скетч (програма) для ШІМ регулюванням яскравості світіння світлодіоду має наступний вигляд:

```
const int LED=6;
void setup ()
{
pinMode(LED, OUTPUT);
}
void loop()
{
```

for (int i<256; /*** Перший цикл ШІМ-збільшення яскравості для змінної **int** LED при **i=0; i=i+1**, який підключений до ШІМ-піну 6 плати Arduino UNO при виконанні умови **i<256 ***/

```
{
analogWrite(LED, i);
delay(10);
}
```

for (int i=255; i>=0; i=i-1) /*** Другий цикл ШІМ-зменшення яскравості для змінної **int** LED при **i=255; i=i-1**, який підключений до ШІМ-піну 6 плати Arduino UNO при виконанні умови **i>=0 ***/

```
{
analogWrite(LED, i);
delay(10);
}
}
```

В програмі між і-тими ШІМ-перемиканнями LED час запізнення складає 10 мс **delay(10),** що менше часу адаптації очей людини до вимкненого стану світлодіоду і тому зміна яскравості світіння світлодіоду сприймається людиною як безперервне зміна наростання та спаду яскравості. Циклові зміни напругі на світлодіоді на кожному ШІМ-кроці складає величину 2V/256=0.0078V=7.8mV.

Команду **analogWrite(LED, i)** має пояснення, яке пов'язане з рис. 4.24. При збільшенні яскравості світлодіоду LED ШІМ-значенні **i=0** шпаруватість імпульсів S=0 і сигнал має низький рівень напруги (**LOW**), тобто **Uвих=0V** (рис. 4.32,а) і світлодіод вимкнений.

При ШІМ-значенні **i=64** шпаруватість імпульсів S=4 і сигнал знаходиться на високому рівні напруги +5V (HIGH) на протязі 25% часу повного періоду T, тобто Uвих=5V/4=1.25V (рис. 4.32,б) і з'являється слабка яскравість світлодіоду LED.

При ШІМ-значенні **i=128** шпаруватість імпульсів S=2 і сигнал знаходиться на високому рівні напруги +5V (HIGH) на протязі 50% часу повного періоду T, тобто Uвих=5V/2=2.5V (рис. 4.32,в) і яскравість світлодіоду LED збільшується.

При ШІМ-значенні **i=192** шпаруватість імпульсів S=1.33 і сигнал знаходиться на високому рівні напруги +5V (HIGH) на протязі 75% часу повного періоду T, тобто Uвих=5V/1.33=3.76V (рис. 4.32,г) і яскравість світлодіоду LED ще збільшується.

При ШІМ-значенні **i=256** шпаруватість імпульсів відсутня і сигнал має високий рівень напруги (**HIGN**), тобто Uвих=5V (рис. 4.32,д) і яскравість світлодіоду LED максимальна.

При зменшенні яскравості світлодіоду LED від максимальної до нуля ШІМ-значення зменшується при зміні від **i=255 до i=0** і рис. 4.32 треба розглядати у зворотному напрямку від рис.4.32,д до рис. 4.32,а.

ШІМ-сигнали можна отримати і без контролера, наприклад, при використанні поширеної мікросхеми **NE555 (**рис.4.33) або модуля генератора на базі таймера **NE555** (рис. 4.34).

На рис. 4.33 представлено електричну принципову схему Ш І М-генератора двигуна **М** постійного струму на мікросхемі **NE555**, а на рис. 4.34 – на базі модуля генератора на базі таймера **NE555.**

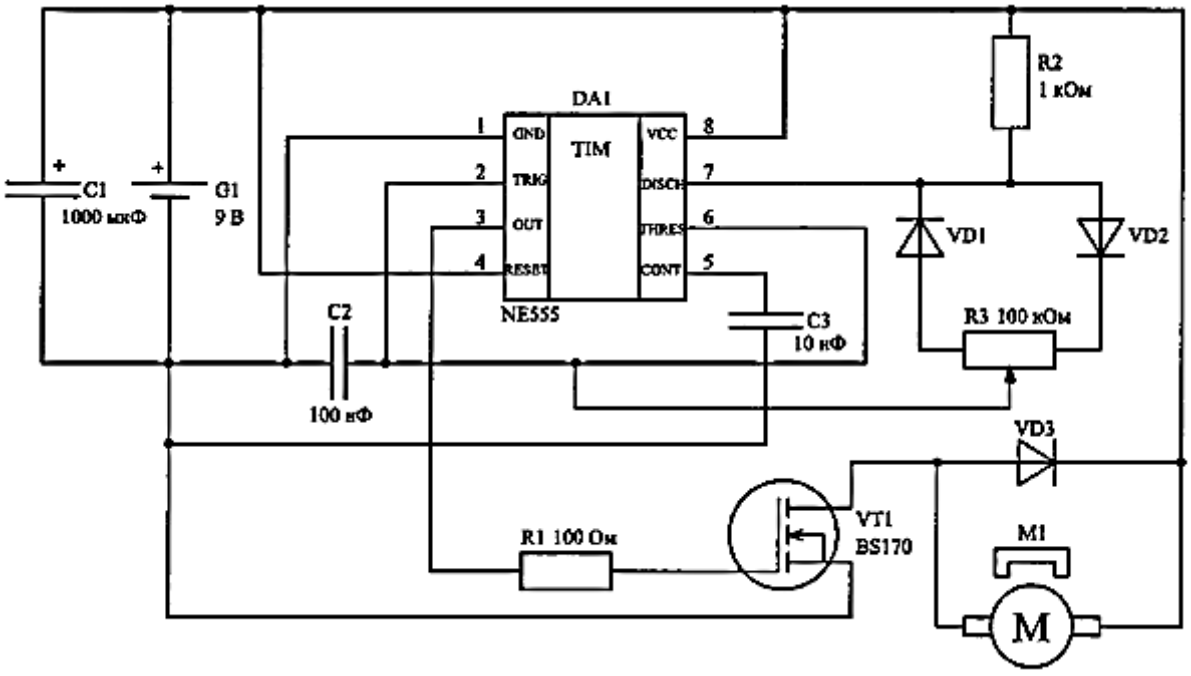


Рис. 4.33. Електрична схема ШІМ-регулятора оборотів двигуна М на мікросхемі таймера **NE555**

Для створення ШІМ-регулятора оборотів двигуна, застосовується нестабільний режим роботи таймера NE555 – постійна генерація імпульсів на виході. Резистори R2 та R3 і конденсатор C2 регулюють шпаруватість імпульсів. Змінний резистор R3 регулює тривалість сигналу високого рівня (**HIGH**). Діоди VD1 і VD2 призначені для того, щоб конденсатор C2, що впливає на вихідну частоту, постійно перезаряджається через різні плечі змінного резистора R3. Заряджається цей конденсатора по ланцюгу «+» джерела живлення, резистор R2, діод VD2 і праве плече змінного резистора. Розряджається конденсатор C2 по ланцюгу - ліве плече змінного резистора, діод VD1 та контакт №7 мікросхеми DA1. Праве і леве плече змінного резистора мають різний електричний опір і тому по різному впливають на час заряду і розряду конденсатора C2. Це забезпечує можливість регулювання шпаруватості сигналу на виході, яким керується швидкість обертання двигуна M постійного струму.

Загальний вигляд поширеного в мехатроніці модуля генератора на базі таймера NE555 і його електрична принципова схема наведені на рис. 4.34.

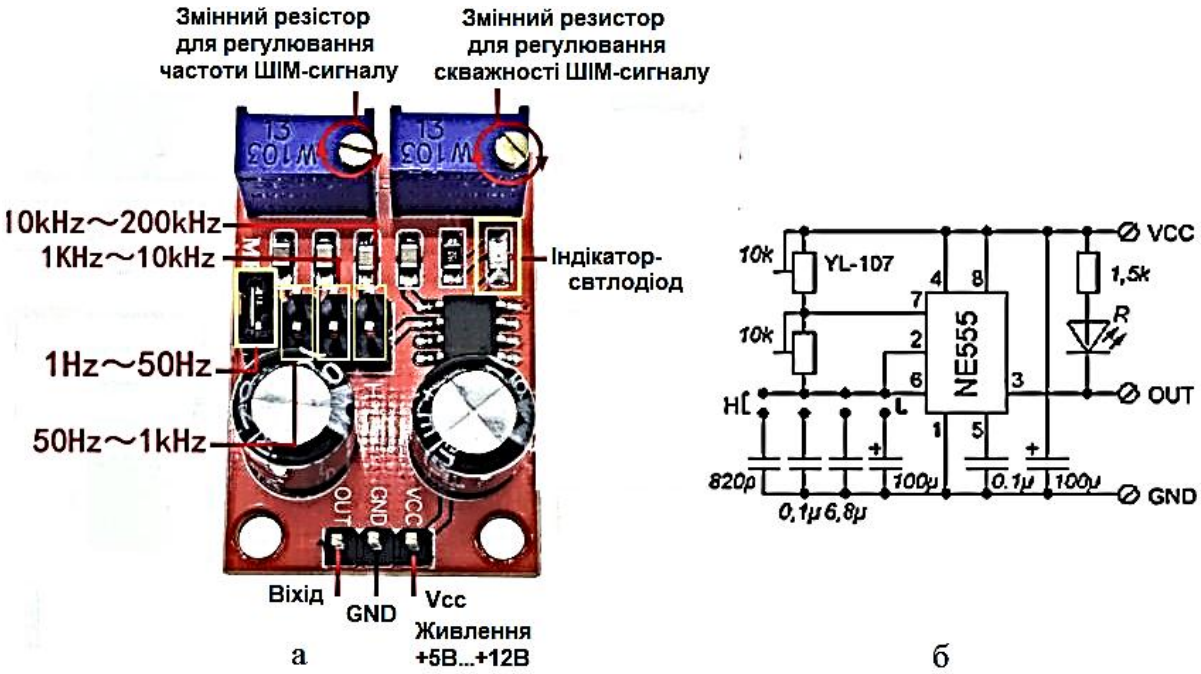


Рис. 4.34. Загальний вигляд модуля генератора на базі таймера NE555 (а) і електрична принципова схема (б) модуля

Контрольні питання до розділу 4:

1. Надайте визначення електропривода CNC-машин.

2. Що таке ***Clutch Motor*** і як регулюється частота обертання головного валу швейної машини ?

3. Які різниця між *Clutch Motor* і ***сервоприводом*** швейних машин ?

4. Що таке потужність крокового електродвигуна і як вона розраховується ?

5. Що таке момент інерції приведений до головного валу технологічної машини і які параметри впливають на вибір автоматизованого приводу технологічної машини або верстату машинобудування ?

6. Які крокові двигуни, які режими їх роботи використовують в схемах мехатроніки і який принцип роботи крокового приводу?

7. Що називається сервоприводом і серводвигуном ?

8. Що таке компіляція і як вона здійснюється у Arduino IDE ?

9. Що позначає скорочена назва PWM і які піни Arduino UNO призначені для реалізації PWM ?

10. Як розраховується шпаруватість імпульсів ?

11. Для чого і як програмується ШІМ ?

12. Яке призначення платформи Arduino ?

13.Які особливості візуального програмування і що таке компіляція привести на конкретному прикладі ?

14.Чем відрізняється за призначенням мікроконтролер від мікропроцесора ?

15.Що відбувається при активізації меню «Інструменти», меню «Файл» і меню «Скетч» ?

Приклади тестових питань для закріплення знань по розділу 4 :

Тест 1. Виберіть помилкове твердження:

• паяльні пари дратують і їх не повинні вдихати ;

• паяльні пари містять свинець ;

• паяльні пари містять флюс, який є головним чином з деревинної смоли ;

• пайка повинна проводитися в вентильованої зоні.

Правильно. Свинець має набагато більш високу температуру плавлення, ніж може досягти ваш паяльник. Але пари пайків і раніше є незначно небезпечними, і вплив має бути обмежена.

Тест 2.

Піни на платі Arduino Uno з написом Digital **0…13** :

• можуть використовуватися тільки як вхідні дані ;

• можуть використовуватися тільки як виходи ;

• можуть використовуватися як входи або виходи ;

• всі здатні до PWM.

Правильно. Можуть використовуватися як входи або виходи. Для **PWM** (широтно-імпульсної модуляції) можуть використовуватися тільки контакти **3, 5, 6, 9, 10** і **11** (позначені на платі контролера символом **~**).

Тест 3.

Що робить функція «**`void loop()`**» ?

• назавжди за циклює ваш код ;

• порівнює одне значення з іншим ;

• чекає введення користувача ;

• повторює фрагмент коду певну кількість разів до вимкнення контролера.

Правильно. Повторює фрагмент коду кількість разів до вимкнення контролера.

Тест 4.

Ви хочете створити проект зі стрічкою світлодіодів 25 штук **NeoPixels** з половиною яскравості. Яке джерело живлення ви повинні використовувати?

• USB-порт комп'ютера

• Акумулятор 500 mAh або адаптер змінного струму

• 1.5 А акумулятор або адаптер змінного струму

Правильно. 1.5 А акумулятор або адаптер змінного струму, тому що кожен світлодіод споживає 40 мА (половина від макс. 80 mA для світлодіодів типу RGBW NeoPixel) і вам буде потрібно як мінімум (40x25=1000 мAh), плюс мікроконтролер або будь-які інші активні компоненти.

5. РОЗРАХУНОК МЕХАТРОННИХ МОДУЛІВ ПРОГРАМОВАНИХ ПЕРЕМІЩЕНЬ З КРОКОВИМ ПРИВОДОМ

Розрахунок модулів програмованих переміщень CNC-машин легкої промисловості і CNC-верстатів машинобудування це, насамперед, розрахунок крутного моменту крокового приводу з урахуванням маса-інерційних параметрів і сил тертя з боку об'єктів програмованих переміщень з наступним вибором драйверів для обраної потужності приводу і програмування траєкторії руху робочого інструменту та/або заготовки з різних матеріалів, в тому числі і матеріалів цільового призначення легкої промисловості .

5.1. **Розрахунок мехатронного модуля програмованих переміщень з тросової передачею**

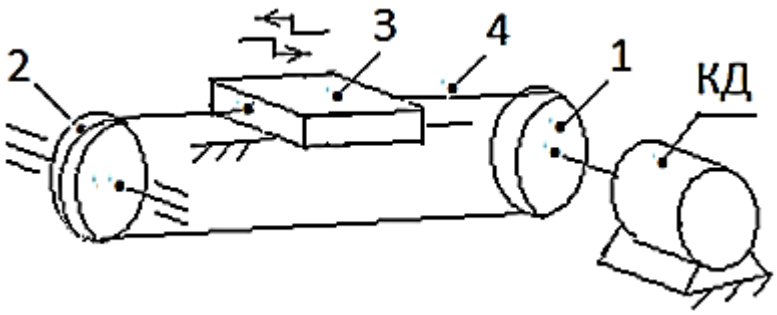


Рис. 5.1. Кінематична 3D-схема модуля програмованих переміщень з тросової передачею

В модулі програмованих переміщень з тросової передачею (рис. 5.1.) на валу крокового двигуна КД закріплений шків 1 радіусом R_1 і моментом інерції **J**. За допомогою невагомого і нерозтяжного тросу у

горизонтальної площині переміщується каретка 3 масою **m** та з коефіцієнтом тертя **f**.

Завдання 1. Необхідно визначити величину необхідного крутного моменту $M_{кд}$ крокового двигуна, який потрібно прикласти до шківа 1, щоб його кутове прискорення дорівнювалося заданої величини **ε** для переміщення каретки із швидкістю **V.**

Позначення, які використовуються в розрахунках:

$R_1 = R_2 = R$ – радіус шківів, м;

$m_1 = m_2 = m$ – маса шківів 1 і 2, кг;

$J_1 = J_2 = J$ – момент інерції шківів, кг·м^2;

m_3 – маса каретки, кг;

g = 9,8 м/с^2 – прискорення вільного руху ;

ε – необхідне прискорення , 1/с^2 ;

V – швидкість каретки, м/с ;

f – коефіцієнт тертя каретки по направляючої ;

φ – кутовий крок двигуна, градуси (з врахуванням режиму дроблення кроку) ;

ν – частота кроків двигуна 1/с;

ω – кутова швидкість, 1/с.

1.1. Визначаємо кінетичну енергію системи:

$$E = \frac{1}{2}\left(m_3 \frac{R^2}{\omega^2} + 2 \cdot J + J_{кд}\right)\omega^2 = 0{,}5\, J_{пр}\, \omega^2 \ , \qquad (5.1)$$

де $J = \frac{1}{2} m R^2$ – момент інерції одного шківа ;

$J_{кд}$ – момент інерції ротора крокового двигуна;

$J_{пр} = m_3 R^2 + 2J + J_{кд}$ – приведений момент інерції.

1.2. Визначаємо похідну за часом від кінетичної енергії

$$\frac{dE}{dt} = J_{пр} \cdot \omega \cdot \varepsilon \ . \qquad (5.2)$$

1.3. Визначаємо потужність зовнішніх сил в системі:

1.3.1. Потужність на подолання сили ваги каретки : $P_{mg} = - m_3 g\, V$.

1.3.2. Потужність на подолання сили тертя: $P_{mp} = - F_{mp} V$,

де $F_{mp} = m_3 g\, f$ – сила тертя.

1.3.3. Потужність крутного моменту: $P_M = \mathrm{M}_{кд}\omega$.

Суму потужності всіх зовнішніх сил визначаємо за виразом:

$$\sum P_i = \mathrm{M}_{кд}\omega - (m_3 g\, V + F_{mp} V) \qquad (5.3)$$

і врахуючи, що V = ω·R отримуємо:

$$\sum P_i = (\mathrm{M}_{кд} - (m_3\, R + F_{mp}\, R))\omega \qquad (5.4)$$

1.4. Похідна від кінетичної енергії за часом $\frac{dE}{dt}$ визначається потужністю всіх зовнішніх сил $\sum \mathrm{P_i}$, тому можна прирівняти праві частини виразів (5.2) і (5.4):

$$J_{пр}\,.\omega \cdot \varepsilon = [\mathrm{M}_{кд} - (m_3 \cdot g \cdot R + F_{mp} \cdot R)] \cdot \omega\ . \qquad (5.5)$$

1.5. Кутова швидкість ω і частота *ν* відпрацювання кроків двигуна знаходяться у наступному співвідношенні:

$$\omega = \frac{H \cdot \nu}{2\pi}, \qquad (5.6)$$

де $H = \frac{360}{\varphi}$ – кількість кроків (мікрокроків) одного обороту ротора крокового двигуна.

З виразу (5.5) отримуємо величину моменту $\mathrm{M}_{кд}$, який потрібно прикласти до шківа 1, щоб його кутове прискорення дорівнювалось ε:

$$\mathrm{M}_{кд} = J_{пр} \cdot \varepsilon + (m_3 \cdot g + F_{mp}) \cdot R = (m_3 \cdot R^2 + 2\, J + J_{кд})\, \varepsilon + (m_3 \cdot g + F_{mp}) \cdot R. \qquad (5.7)$$

Швидкість переміщення каретки з урахування (5.6):

$$V = \omega \cdot R = \frac{H \cdot \nu \cdot R}{2\pi} \qquad (5.8)$$

Частота $\boldsymbol{\nu}$ кроків, яка необхідна для переміщення каретки із швидкістю V

$$\nu = \frac{2\pi \cdot V}{H \cdot R} \qquad (5.9)$$

Наприклад, при кроці $\varphi = 0{,}9^0$ і $H = 400$ кроків один оборот $\nu = \frac{\pi \cdot V}{200 \cdot R}$.

З урахуванням (5.8) і (5.9) ротор крокового двигуна, який починає відпрацювання кроків з частотою $\boldsymbol{\nu}$ на першому кроці рухається з прискоренням:

$$\boldsymbol{\varepsilon = \nu \cdot \omega} = \frac{\nu^2 \cdot \mathrm{H}}{2\pi} = \frac{2\pi \cdot V^2}{H \cdot R^2} \qquad (5.10)$$

Необхідний крутний момент (5.7) на валу крокового двигуна, який зможе переміщувати вантаж із швидкістю V визначаємо з урахуванням виразу (5.10) за формулою (5.11) або за формулою (5.12):

$$\mathrm{M_{кд}} = (m_3 R^2 + 2J + J_{кд}) \frac{\nu^2 \cdot \mathrm{H}}{2\pi} + (m_3 g + F_{тр}) \cdot R \qquad (5.11)$$

$$\mathrm{M_{кд}} = (m_3 R^2 + 2J + J_{кд}) \frac{2\pi \cdot V^2}{H \cdot R^2} + (m_3 g + F_{тр}) \cdot R \qquad (5.12)$$

5.2. **Розрахунок мехатронного модуля програмованих переміщень із зубчаста-пасовою передачею в мехатронної системі**

Кінематична схема модуля програмованих переміщень із зубчаста-пасовою передачею (рис.5.2.) складається з крокового двигуна розташованого вертикально, на валу ротора якого змонтований зубчасте колесо 1 з моментом інерції J_1. Зубчасте-пасова передач 4 кінематичне з'єднує зубчасте колесо 1 із зубчастим колесом 2 з моментом інерції J_2. На одної гілці пасової передачі закріплена каретка 3 з можливістю зворотно-поступових програмованих переміщень між зубчастими колесами. Кінцеві вимикачі крайніх положень каретки із заготовкою або каретки у вигляді п'ялець з текстильним матеріалом на схемі не зображені. Момент тертя в підшипниках $M_{тр}$.

Завдання 2. Визначити величину крутного моменту $M_{кд}$ крокового двигуна, який потрібно прикласти до зубчастого колеса 1 щоб його кутове прискорення дорівнювалось **ε** .

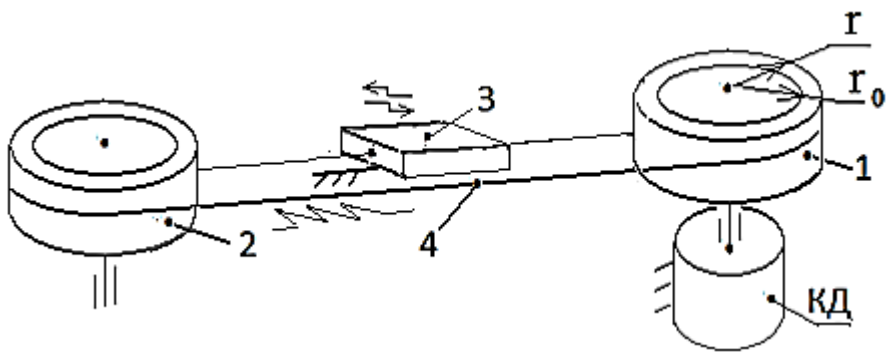


Рис. 5.2. Кінематична 3D-схема модуля програмованих переміщень каретки 3 із зубчасте-пасовою передачею

Позначення, які використовуються в розрахунках:

r – радіус зовнішній зубчастих колес 1 і 2 ;

r_0 – радіус внутрішній зубчастих колес 1 і 2 ;

L – товщина зубчастих колес ;

$m_1 = m_2$ – маса зубчастого колеса;

m_3 – маса каретки ;

$J_1=J_2=J= 0{,}5\, m_{зк}(r^2 - r_0^2)$ – момент інерції зубчастого колеса ;

$J_{кд}$ – момент інерції ротора двигуна;

ω – кутова швидкість.

2.1. Визначаємо кінетичну енергію системи:

$$E = 0{,}5\ (J_{кд} + J_1 + J_2 + \frac{m_з V^2}{\omega^2})\ \omega^2\ , \qquad (5.13)$$

де J_1 та J_2 - моменти інерції зубчастих колес 1 і 2 ;

$J_{кд}$ - момент інерції ротора крокового двигуна;

2.2. Визначаємо похідну від кінетичної енергії за часом:

$$\frac{dE}{dt} = (J_{кд} + J_1 + J_{2.})\cdot\omega\cdot\varepsilon + m_з\cdot V\cdot a \qquad (5.14)$$

2.3. Потужність зовнішніх сил в системі:

Потужність на подолання моменту від сили тертя: $P_{тр} = M_{тр}\cdot\omega$.

Потужність крутного моменту: $P_M = M_{кд}\cdot\omega$.

Суму потужності всіх сил визначаємо за виразом:

$$\sum P_i = (M_{кд} - M_{тр})\cdot\omega \qquad (5.15)$$

2.4. Похідна від кінетичної енергії за часом $\frac{dE}{dt}$ визначається потужністю всіх зовнішніх сил $\sum P_i$, тому можна прирівняти праві частини виразів (5.14) та (5.15):

$$(J_{кд} + 2J)\,\omega\cdot\varepsilon + m_з\cdot V\cdot a = (M_{кд} - M_{тр})\,\omega. \qquad (5.16)$$

З виразу (5.16) визначаємо величину крутного моменту крокового двигуна:

$$M_{кд} = (J_{кд} + 2J)\,\varepsilon + \frac{m_з\cdot V\cdot a}{\omega} + M_{тр} \qquad (5.17)$$

5.3. **Розрахунок мехатронного модуля програмованих переміщень з рейковою передачею в мехатронної системі**

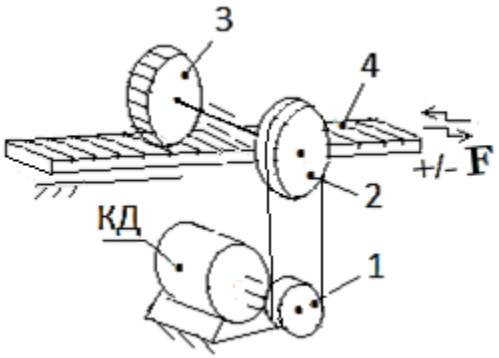


Рис. 5.3. Кінематична 3D-схема модуля програмованих переміщень з рейковою передачею

Модуль містить кроковий привод із зубчасте-пасовою передачею і рейкову передачу. На валу двигуна (рис. 5.3) закріплена зубчасте колесо 1, яке зубчасте-пасовою передачею передає рух зубчастому колесу 2 і шестерні 3 зубчастої рейки 4. Коефіцієнт тертя в рейковій передачі $f_{тр}$.

Завдання 3. Визначити величину крутного моменту $M_{кд}$, який потрібно прикласти до шестерні, щоб кутове прискорення зубчастої рейки було рівно **а**.

Позначення, які використовуються в розрахунках:
R_1 і R_2 – радіус зубчастих колес 1 і 2, м ;
m_1 і J_1 та m_2 J_2 – маса і момент інерції зубчастих колес, кг, кг·м^2;
R_3 – радіус шестерні 3, м ;
m_3 , J_3 – маса і момент інерції шестерні, кг, кг·м^2;
m_4 , J_4 – маса і момент інерції зубчастої рейки, кг, кг·м^2 ;
$J_{кд}$ – момент інерції ротора крокового двигуна, кг·м^2;
F – тягнуче каретку зусилля, Н ;
L – довжина зубчастої рейки, м ;
i – передаточне відношення ;
g = 9,81 м/с^2, прискорення вільного руху ;

Визначення обертаючого моменту $M_{кд}$ виконуємо в наступному порядку:

3.1. Розраховуємо маса-інерційні параметри системи:

$J_1 = 0{,}5\, m_1 \cdot R_1^2$ – момент інерції зубчастого колеса 1;

$J_2 = 0{,}5\, m_2 \cdot R_2^2$ – момент інерції зубчастого колеса 2;

$J_3 = 0{,}5\, m_3 \cdot R_3^2$ – момент інерції шестерні 3;

$J_4 = m_4 \cdot R_3^2$ – момент інерції зубчастої рейки 4.

$J_{сум} = J_{кд} + J_1 + J_2 + J_3 + J_4$ – сумарний момент інерції системи.

3.2. Визначаємо кінетичну енергію системи:

$$E = 0{,}5\,(J_{кд}\,\omega^2 + J_1\,\omega^2 + i \cdot J_2\,\omega^2 + i \cdot J_3\,\omega^2 + m_4\,V^2) = 0{,}5\, m_{пр}\, V^2, \qquad (5.18)$$

де $m_{пр} = \frac{J_{кд} + J_1 + i \cdot J_2 + i \cdot J_3}{R_2^2} + m_4$ – приведена маса системи.

3.3. Визначаємо похідну від кінетичної енергії за часом:

$$\frac{dE}{dt} = m_{пр} \cdot V \cdot a \qquad (5.19)$$

3.4. Сумарна потужність всіх сил:

$$\sum P_i = [M_{кд} - f_{тр}(m_4 \cdot g \cdot R_3)] \cdot \frac{V}{R_3} \qquad (5.20)$$

3.5. Похідна від кінетичної енергії за часом $\frac{dE}{dt}$ визначається потужністю всіх зовнішніх сил $\sum P_i$, тому можна прирівняти праві частини виразів (5.19) і (5.20):

$$m_{пр} \cdot V \cdot a = [M_{кд} - f_{тр}(m_4 \cdot g \cdot R_3)] \cdot \frac{V}{R_3} \qquad (5.21)$$

3.6. Величину обертаючого моменту крокового двигуна визначаємо з виразу (5.21):

$$M_{кд} = (m_{пр} \cdot a + f_{тр} \cdot m_4 \cdot g) \cdot R_3 \qquad (5.22)$$

5.4. **Розрахунок мехатронного модуля програмованих переміщень з кулько-гвинтовою передачею CNC-машин і CNC-верстатів**

На вітчизняних підприємствах легкої і текстильної промисловості використовується технологічні машини закордонних фірм та американські, англійські, японські технології машинобудування з *дюймовою системою нарізів*. Це кріплення і передачі з наступними різьбленнями двох типів:

UNF (*англ.* **UN**ified **F**ine) – точне різьблення з дрібним кроком;

UNC (*англ.***UN**ified **C**oarse) – грубе різьблення з великим кроком.

Розрахунок приводу з кроковим або вентильним двигуном доцільно починати з вирішення наступних двох завдань деталей машин:

Завдання А – визначення величини переміщення по осі гвинта з дюймовим нарізом веденої ланки (гайки) в мм на 1 оборот гвинта;

Завдання Б – визначення величини кута повороту (оборотів) гвинта з дюймовим нарізом на 1 мм переміщення по осі веденої ланки веденої ланки (гайки).

При використанні пари «гвинт-гайка» з метричним нарізом, наприклад, для нарізу М*1 – за 1 оберт гвинта гайка переміщується по осі на 1 мм і навпаки для програмування переміщення гайка на 1 мм по осі ротор потрібно запрограмувати 1 оберт (кут 2π радіан або 360^0) двигуна і гвинта, що з'єднані напряму без механічних передач.

При дюймових нарізах (дюйм *англ.* **inch**) позначення нарізу $0{,}125''$ для однозаходного гвинта це крок гвинта, а саме відстань перемыщення гайки по осі на 1/8 дюйма = 0,125 дюйма = (0,125*25,4 мм) = 3,175 мм.

З урахування кількості заходів гвинта крок k гвинта з дюймовим нарізом визначається за формулою:

$$k = \frac{z}{n_{inch}} . \frac{N_s}{N_m}, \qquad (5.22)$$

де z – кількість заходів нарізу;

n_{inch} – кількість витків на 1 дюйм (25,4 мм);

N_s – кількість зубців шестерні (зірочки) на валу гвинта;

N_m – кількість зубців шестерні (зірочки) на валу електродвигуна.

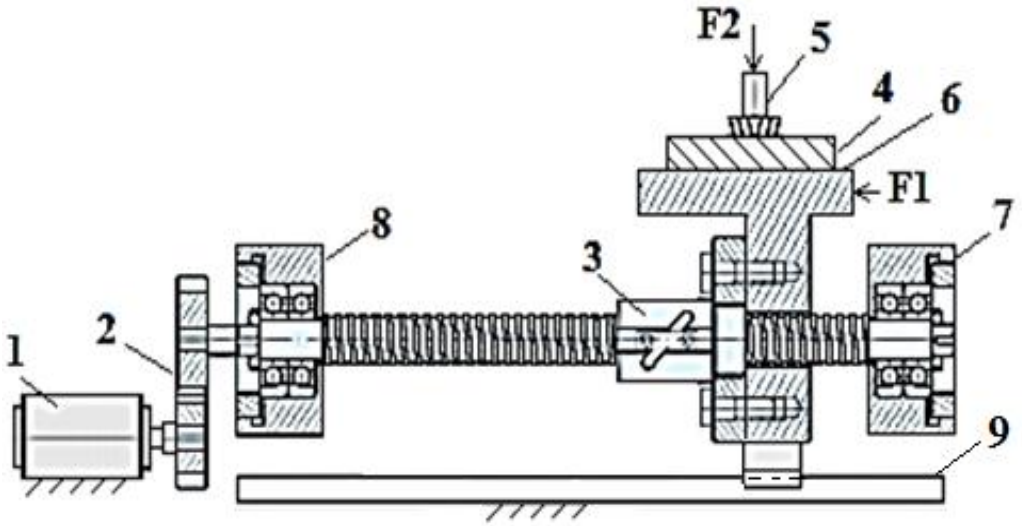


Рис. 5.4. Конструктивно-кінематична схема модуля програмованих переміщень з кулько-гвинтовою передачею CNC-верстатів

На рис. 5.4 прийняти наступні позначення елементів, які враховані в розрахунку модуля програмованих переміщень: 1 – сервопривод на засадах крокового двигуна; 2 – редуктор; 3 – кулько-гвинтова передача; 4 – заготовка; 5 – робочий інструмент (торцева фреза); 6 – каретка (стіл); 7 і 8 – упорні дворядні підшипники; 9 – направляча

Розглянемо порядок розрахунків на прикладі фрезерного CNC-верстату.

Для схеми на рис. 5.4 задані наступні вихідни дані:

вага каретки 6: $W_6 = 200$ кгс ;

вага заготівля 4: $W_4 = 100$ кгс ;

коефіцієнт терття опор кочення для кулько-гвинтової передачі μ = 0,02 ;

прийнятий наступний цикл роботи і навантаження приводу (табл. 5.1):

Таблиця 5.1

Діяче осьове робоче навантаження на гвинт, *кгс*	Частота обертання, *об/хв*	Час дії навантаження, *%*
$F_1 = 100$	$n_1 = 500$	$t_1 = 20$
$F_2 = 300$	$n_2 = 100$	$t_2 = 50$
$F_3 = 500$	$n_3 = 50$	$t_3 = 30$

прискорення ε = 100 рад/с;

діаметр і довжина ротора (якоря) двигуна 1: $d_1 = 50$ мм; $l_1 = 200$ мм;

діаметр і кількість зубців ведучей шестерні трансмісії: $d_2 = 80$ мм; $z_1 = 30$;

діаметр і кількість зубців веденої шестерні трансмісії: $d_3 = 240$ мм; $z_2 = 90$;

товщина шестерен: b = 20 мм;

параметри КГП: діаметр гвинта $d_г = 50$ мм ;

довжина гвинта $l_г = 1200$ мм ;

крок гвинта $p_г = 10$ мм ;

вага гвинта $W_г = 18$ кгс ;

обертаючий момент для підшипникового вузла $M_2 = 10$ кгс · мм

аксіальне навантаження попереднього натягу, при якому відсутній зазор в кінематичної парі гвинт-кульки гайки: $F_0 = 300$ кгс ;

коефіцієнт корисної дії (к.к.д) або механічна ефективність КГП $\eta_1 = 0{,}80$

коефіцієнт моменту попереднього натягу $k_p = 0{,}2$;

момент для підшипникового вузла $M_3 = 10$ кгс · мм.

Порядок розрахунків:

Визначаємо момент M_0 [кгс·мм] модуля програмованих переміщень з кулько-гвинтовою передачею *при заданих умовах роботи при постійної швидкості (ε = 0):*

1. З урахування n_i і t_i робочих умов роботи приводу (табл. 5.1) середнє число оборотів в хвилину дорівнюється:

$$n_c = n_1 \cdot \frac{t_1}{100} \cdot \frac{n_1}{n_c} + n_2 \cdot \frac{t_2}{100} + n_3 \cdot \frac{t_3}{100} = 500 \cdot \frac{20}{100} + 100 \cdot \frac{50}{100} + 50 \cdot \frac{30}{100} = 165 \frac{\text{об}}{\text{хв}}$$

2. Середнє осьове (аксіальне) робоче навантаження на гвинт при постійної швидкості визначаємо і розраховуємо за формулою з урахуванням F_i і t_i (табл. 5.1):

$$F_c = \sqrt[3]{F_1^3 \cdot \frac{t_1}{100} \cdot \frac{n_1}{n_c} + F_2^3 \cdot \frac{t_2}{100} \cdot \frac{n_2}{n_c} + F_3^3 \cdot \frac{t_3}{100} \cdot \frac{n_3}{n_c}} =$$

$$= \sqrt[3]{100^3 \cdot \frac{20}{100} \cdot \frac{500}{165} + 300^3 \cdot \frac{50}{100} \cdot \frac{100}{165} + 500^3 \cdot \frac{30}{100} \cdot \frac{50}{165}} = 272 \text{ кгс}$$

3. Врахування додаткового осьового навантаження на гвинт від попереднього натягу в кінематичної пару гвинт-гайка (при відсутності зазору):

$$F_{\text{п}} = \frac{F_0}{2{,}8} = \frac{300}{2{,}8} = 107 \text{ кгс}$$

4. Загальне осьове навантаження на гвинт з урахіванням сил терття:

$$F_{\sum} = F_{\text{с}} + \mu(W_6 + W_4) = 272 + 0{,}02(200 + 100) = 278 \text{ кгс}$$

5. Момент від навантаження з урахуванням виразу по п.4.4 , крока гвинта $p_{\text{г}} = 10$ мм і к.к.д. $\eta_1 = 0{,}80$:

$$M_1 = F_{\sum} \frac{p_{\text{г}}}{2\pi \cdot \eta_1} = 278 \cdot \frac{10}{2\pi \cdot 0{,}80} = 553 \text{ кгс} \cdot \text{мм}$$

6. Момент з урахуваннням виразу п.4.3:

$$M_2 = F_{\text{п}} \frac{k_p \cdot p_{\text{г}}}{2\pi} = 107 \cdot \frac{0{,}2 \cdot 10}{2\pi} = 34 \text{ кгс} \cdot \text{мм}$$

7. Загальний момент M_0 при постійної швидкості з урахуванням виразів по п. 4.5 , п. 4.6, моменту M_3 і трансмісії дорівнюється:

$$M_0 = (M_1 + M_2 + M_3) \cdot \frac{z_1}{z_2} = (553 + 34 + 10) \cdot \frac{30}{90} = 199 \text{ кгс} \cdot \text{мм}$$

Визначаємо маса-інерційні параметри ланок приводу:

8. Момент інерції ротора(якоря) крокового двигуна визначаємо за формулою:

$$J_1 = \frac{1}{2} m_{\text{р}} R_{\text{р}}^2 = \frac{\pi \cdot R_{\text{р}}^4 \cdot \rho \cdot l_1}{2 \cdot g} = \frac{3{,}14 \cdot 25^4 \cdot 7{,}8 \cdot 10^{-6} \cdot 200}{2 \cdot 9800} = 0{,}1 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2,$$

де $m_{\text{р}} = \pi \cdot R_{\text{р}}^2 \cdot l_1 \cdot \rho$ – маса ротора,

$\rho = \frac{7{,}8 \cdot 10^3}{10^9} = 7{,}8 \cdot 10^{-6}$ $\frac{\text{кг}}{\text{мм}^3}$ – питома щільність матеріалу ротора,

$g = 9800\ {}^{\text{мм}}/_{\text{с}^2}$ – прискорення выльного падіння ваги.

9. Момент інерції ведучей шестерні масою m_2 трансмісії:

$$J_2 = \frac{1}{2} m_2 R_2^2 = \frac{1}{2 \cdot 9800} \cdot 3{,}14 \cdot 7{,}8 \cdot 10^{-6} \cdot \left(\frac{80}{2}\right)^4 \cdot 20 = 0{,}064 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2$$

10. Момент інерції веденої шестерні масою m_3трансмісії:

$$J_3 = \frac{1}{2} m_3 R_3^2 = \frac{1}{2 \cdot 9800} \cdot 3{,}14 \cdot 7{,}8 \cdot 10^{-6} \cdot \left(\frac{240}{2}\right)^4 \cdot 20 = 5{,}18 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2$$

11. З урахуванням передаткового відношення та виразів по п.4.9 і по п.4.10 момент інерції трансмісії дорівнюється:

$$J_\text{т} = J_2 + J_3 \cdot \left(\frac{z_1}{z_2}\right)^2 = 0{,}064 + 5{,}18 \cdot \left(\frac{30}{90}\right)^2 = 0{,}64 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2$$

12. Момент інерції гвинта:

$$J_\text{г} = \frac{1}{2} \cdot \frac{W_\text{г}}{g} \cdot \left(\frac{d_\text{г}}{2}\right)^2 \cdot \left(\frac{z_1}{z_2}\right)^2 = \frac{1}{2 \cdot 9800} \cdot 18 \cdot \left(\frac{50}{2}\right)^2 \cdot \left(\frac{30}{90}\right)^2 = 0{,}064 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2$$

13. Момент інерції вага каретки W_6 і ваги заготівлі W_4 обертанні гвинта:

$$J_{64} = \frac{W_6 + W_4}{g} \cdot \left(\frac{p_\text{г}}{2\pi}\right)^2 \cdot \left(\frac{z_1}{z_2}\right)^2 = \frac{200 + 100}{9800} \cdot \left(\frac{10}{6{,}28}\right)^2 \cdot \left(\frac{30}{90}\right)^2 = 0{,}009 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2$$

14. Загальний момент інерції:

$$J_\Sigma = J_1 + J_\text{т} + J_\text{г} + J_{64} = 0{,}1 + 0{,}64 + 0{,}064 + 0{,}009 = 0{,}813 \text{ кгс} \cdot \text{мм} \cdot \text{с}^2$$

15. *Визначаємо номінальний момент* M_Σ для вибору крокового двигуна приводу за розрахованої його потужністі P [кВт] *при заданому прискоренні ε = 100* рад/с з урахуванням виразів по п. 4.14 і по п.4.7:

$$\text{М}_4 = J_\Sigma \cdot \varepsilon = 0{,}813 \cdot 100 = 81{,}3 \text{ кгс} \cdot \text{мм}$$

$$\text{М}_\Sigma = \text{М}_4 + \text{М}_0 = 81{,}3 + 199 = 280{,}3 \text{ кгс} \cdot \text{мм}$$

16. *Визначаємо номінальну потужність P* [кВт] двигуна з урахуванням результату по п.4.15 для М_Σ та з урахуванням запасу надійності с коефіцієнтом запасу рівному 2 для $n_{max} = 1500$ $^{\text{об}}/_{\text{хв}}$ $\left(\omega = \frac{\pi \cdot n_{max}}{30}\right)$:

$$\text{М}_{\Sigma max} = 2 \cdot \text{М}_\Sigma \quad = 2 \cdot 280{,}3 = 560{,}6 \text{ кгс} \cdot \text{мм};$$

$$P = M_{max} \cdot \omega = \frac{\text{М}_{\Sigma max} \cdot \pi \cdot n_{max} \cdot 9{,}8 \cdot 10^{-3}}{30} = \frac{560{,}6 \cdot 1500}{974} = 862 \text{ Вт}$$

Вибір робочого моменту сервоприводу виконується за умовою

$$M_{\text{роб}} \rangle\ 1{,}5 \cdot \text{М}_{\Sigma max}$$

Вибираємо кроковий двигун (сервомотор) з номінальною потужністю P = 950 Вт з номінальним моментом

$\text{М}_\Sigma = 300 \text{ кгс} \cdot \text{мм} = 3 \text{ Н}\cdot\text{м}$ і номінальною швидкістю $n = 2000$ $^{\text{об}}/_{\text{хв}}$,

$\text{М}_{\Sigma max} = 65 \text{ кгс} \cdot \text{см}$ (650 кгс · мм = 6,5 Н·м) та з моментом інерції ротора крокового двигуна, що дорівнюється $J_1 = 0{,}2 \text{ кгс} \cdot \text{мм}^2 = 0{,}2 \cdot 10^{-6}$ кгс·м^2.

В технічної документації фірм-виробників крокових двигунів вказується момент крокових двигунів в стані спокою (утримуючий момент М), який залежить від швидкості роботи приводу. При збільшенні частоти обертання ротора двигуна крутний момент зменшується.

Після вибору крокового двигуна по розрахованої потужності визначається тягове зусилля приводу з урахуванням типу передачі. Наприклад, для модуля програмованих переміщень з кулькове-гвинтової передачею типу 1405 (перші дві цифри – діаметр гвинта 14 мм, останні дві цифри – крок нарізу h=5 мм=0,005 м) та коефіцієнтом корисної дії передачі

η=0,9 для біполярного гібридного крокового двигуна NEMA 23 (1,7А і 6,8 V на фазу та М=9 кг·см=0,9 Н·м) осьове тягове зусилля F розраховуємо за формулою:

$$F = \frac{M \cdot \eta \cdot \pi}{h} = \frac{0{,}9 \cdot 0{,}9 \cdot 3{,}14}{0{,}005} = 507{,}8 \text{ Н}$$

5.5. **Особливості конструкції кулько-гвинтових передач для побудови модулів програмованих переміщень CNC-машин і CNC-верстатів**

Кулько-гвинтові передачі (КГП) складається з гвинта і гайки, об'єднаних разом кульками і механізмом для повернення кульок. Призначені для перетворення обертання гвинта в поступальний рух столу CNC-верстатів, при цьому обертовий момент сервоприводу створює осьове зусилля зі збереженням високого ступеня точності, реверсивності і високої ефективності передачі.

Якщо до крокового приводу або сервоприводу конструктивне додати кулько-гвинтових передачу то отримуємо модуль програмованих прецизійних переміщень робочого органу CNC-машин і CNC-верстатів.

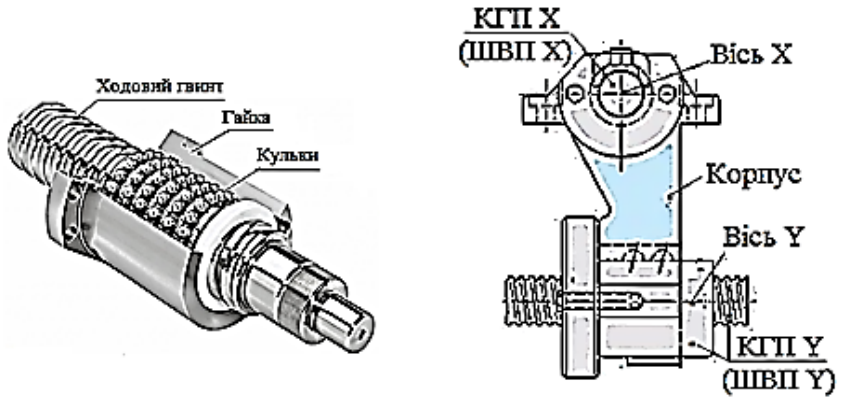


Рис. 5.13. Загальний вигляд КГП і фрагмент конструкції двох модулів програмованих 2-координатних переміщень CNC-машин і CNC-верстатів

Кулько-гвинтові передачі – поширена назва «Шарико-винтовые передачи» (ШВП) використовуються в прецензійному мехатронному обладнанні і верстатах (рис. 5.13) для забезпечення швидкісних характеристик і характеристик точності параметрів. В такому обладнанні

передачі «гвинт-гайка» з трапецеїдальним різьбленням замінюються на КГП, які мають засоби створення перед натягу (попереднього натягу) з метою виключення люфтів в кінематичних парах «кульки-гвинт». В кулько-гвинтових передачах тертя ковзання замінено на тертя кочення.

Галузі застосування модулів з КГП для програмованих переміщень: верстатобудування та машинобудування, обладнання легкої і текстильної промисловості, друкувальне обладнання і обладнання паперової промисловості та приладобудування та інші.

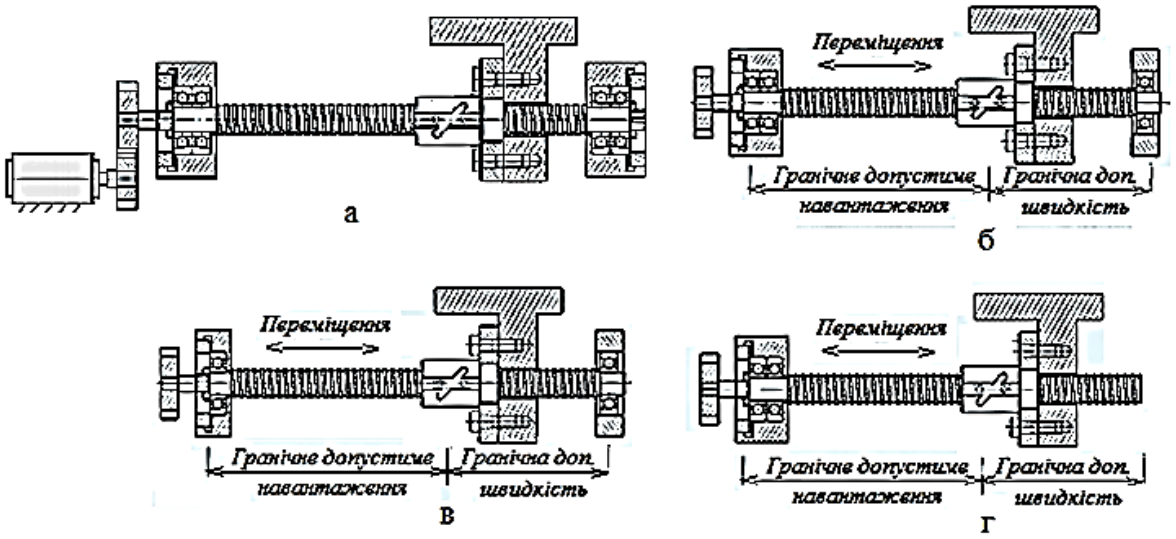


Рис. 5.14. Варіанти способів монтажу КГП: а – з дворядними упорними підшипниками на кінцях гвинта; б – з дворядним упорним і однорядним підтримуючим підшипниками на кінцях гвинта; в - з однорядними підтримуючими підшипниками на кінцях гвинта; г – з дворядним упорним підшипником і вільним монтажем каретки на консолі гвинта

При конструюванні мехатронних систем CNC-верстатів важливо враховувати можливість застосування різних способів монтажу необхідних підшипників на кінцях гвинтів КГП, що визначає жорсткість механічної системи в цілому. Параметр жорсткості обмежує максимальні швидкості і навантаження, які виникають в при роботі в такому обладнанні. Варіанти способів монтажу КГП наведені на рис. 5.14. Вибір варіанту монтажу КГП залежить від приведених до гвинта маса-інерційних параметрів каретки із заготовкою і сил корисного опору при виконанні технологічних операцій.

Кулько-гвинтові передачі мають наступні три типи конструкцій КГП із різними засобами ротації (рециркуляції) сталевих кульок в кінематичної парі «гвинт-гайка».

Тип 1 – це конструкція КГП із зовнішньою рециркуляцією кульок (рис. 5.15,а) що має гвинт і гайку, які утворюють картридж для сталевих кульок із замкненим ланцюгом у вигляді просторої петлі. Безперервна траса для руху кульок утворена гвинтом, гайкою та зворотними трубами, закріпленими ззовні на гайці притискною пластиною.

Тип 2 – це конструкція КГП з внутрішньою рециркуляцією кульок (рис. 5.15,б), у який всередині гайки мається канавка для зворотного повернення кульок на попередню доріжку кочення.

Тип 3 – це конструкція КГП із кінцевою системою повернення кульок (рис. 5.15,в), яка має всередині гайки отвір що виконує зворотне повернення кульок. Такий отвір виконує функцію трубки КГП першого типу.

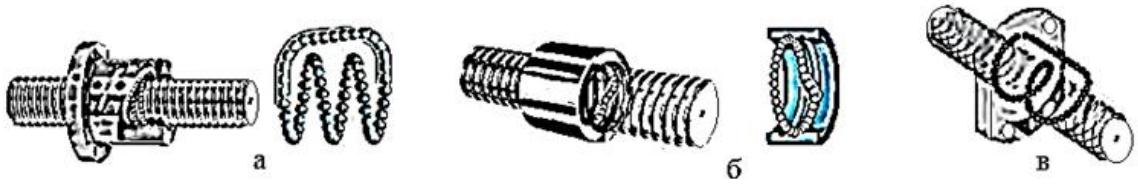


Рис. 5.15. Типи рециркуляції (системи ротації) сталевих кульок кулькове-гвинтових передач: а – зовнішня (*тип 1*); б – внутрішня (*тип 2*); в – з кінцевою системою повернення (*тип 3*)

Різні типи кулько-гвинтових передач мають наступні позначення конструктивних особливостей:

Для КГП із зовнішньою рециркуляцією кульок (*тип 1*):

A – 1,5 обороту на 1 петлю з кульок;

B – 2,5 обороту на 1 петлю з кульок;

C – 3,5 обороту на 1 петлю з кульок;

D – 4,5 обороту на 1 петлю з кульок;

E – 5,5 обертів на 1 петлю з кульок.

Для КГП з внутрішньою рециркуляцією кульок (*тип 2*):

T – 1,0 оборот на 1 петлю з кульок.

Для КГП з кінцевою системою повернення кульок (*тип 3*):

K – 1,0 оборот на 1 петлю з кульок;

U – 2,8 обороту на 1 петлю з кульок (з великим кроком);

S – 1,8 обороту на 1 петлю з кульок (з super великим кроком);

V – 0,7 обороту на 1 петлю з кульок (з extra великим кроком).

Приклади позначенння КГП :

B2: має 2 зовні трубки для 2х петель з кульок. Кожна петля має 2,5 обороту кульок;

T3: має внутрішню систему рециркуляції кульок. Траса складається з 3х петель кульок в гайці. Кожна петля це 1 оберт кульок;

S4: має внутрішню систему рециркуляції кульок з кінцевою системою повернення кульок. Траса складається з 4х петель кульок в гайці. Кожна петля це 1,8 обороту кульок;

K5: має внутрішню систему рециркуляції кульок з кінцевою системою повернення кульок. Траса кульок складається з 5-ти петель кульок в гайці. Кожна петля це 1 оборот кульок.

Контрольні питання до розділу 5.

1. Як визначається приведена до валу крокового двигуна КД кінетична енергія каретки 3 на рис. 5.1 ?

2. Як визначається кінетична енергія ланок з обертовим і зворотно-поступовим рухом ?

3. Для якої мети визначається похідна за часом від кінетичної енергії ?

4. Як визначається величина переміщення по осі гвинта з дюймовим нарізом веденої ланки (гайки) *в мм* на 1 *оборот гвинта* ?

5. Як визначається кут повороту (кількість оборотів) гвинта з дюймовим нарізом *на 1мм* переміщення по осі веденої ланки веденої ланки (гайки) ?

6. В яких машинах легкої промисловості застосовуються модулі програмованих переміщень з кроковим приводом ?

7. Чим відрізняється CNC-машина від машини-автомату ?

8. Що означає скорочення UNF, UNC, CNC-верстат ?

9. Як розрахувати потужність і обрати потрібний сервомотор на засадах крокового двигуна для CNC-верстату ?

10. Що означає скорочення C2 для КГП з рециркуляцією кульок *типу 1* ?

6. МЕХАТРОННІ МОДУЛІ І МЕХАНІЗМИ З ПРОГРАМОВАНИМ ПНЕВМОПРИВОДОМ

6.1. **Виконавчі пневмоцилиндри двосторонньої дії з регульованим гальмуванням у кінці ходу**

Пневмоциліндр двосторонньої дії це пневматичний двигун зворотно-поступальної дії або виконавчий механізм, що перетворює енергію потоку стисненого повітря в енергію поступального руху поршня. На рис. 6.1 наведена конструктивна схема пневмоциліндра подвійної дії.

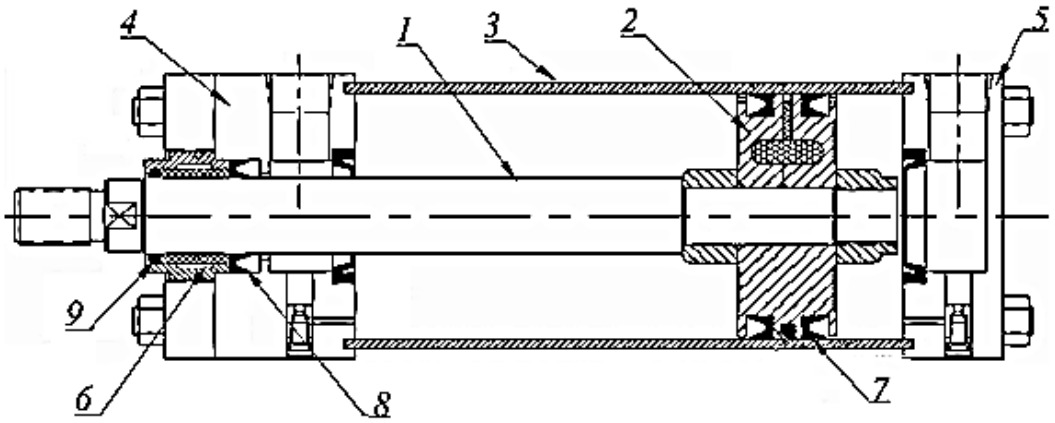


Рис. 6.1. Конструктивна схема пневмоциліндра подвійної дії

На рис. 6.1 прийняті наступні позначення: 1 – шток; 2 – поршень; 3 – гільза; 4 – передня кришка; 5 – задня кришка; 6 – направляюча втулка; 7 – U-подібні ущільнення поршня; 8 – ущільнення штоку; 9 – брудоз'ємник

Пневмоциліндр односторонній дії це також пневматичний двигун що має у штокової порожнині циліндричну пружину (холостий хід при втягуванні поршня) чи у над поршневої порожнині (холостий хід при висуванні поршня). Використовують такі пневмоциліндри в схемах мехатроніки, коли необхідно виконувати прямолінійний рух під навантаженням тільки в одному напрямку.

На рис. 6.2 і рис. 6.3 наведені конструктивні особливості поршнів з різними типами демпфірування та їх якісні характеристики. Тип **P** демпфірування мають поршні з короткою довжиною демпфера, тип **PPV** демпфірування (рис. 6.2) – поршні з подовженою довжиною демпфера і

тип **PPS** демпфірування (рис. 6.3) – ці поршні теж мають подовжену довжиною демпфера але не цей демпфіруючої частині поршня відфрезеровані поздовжні пази, які покращують динаміку руху поршня в крайній положеннях.

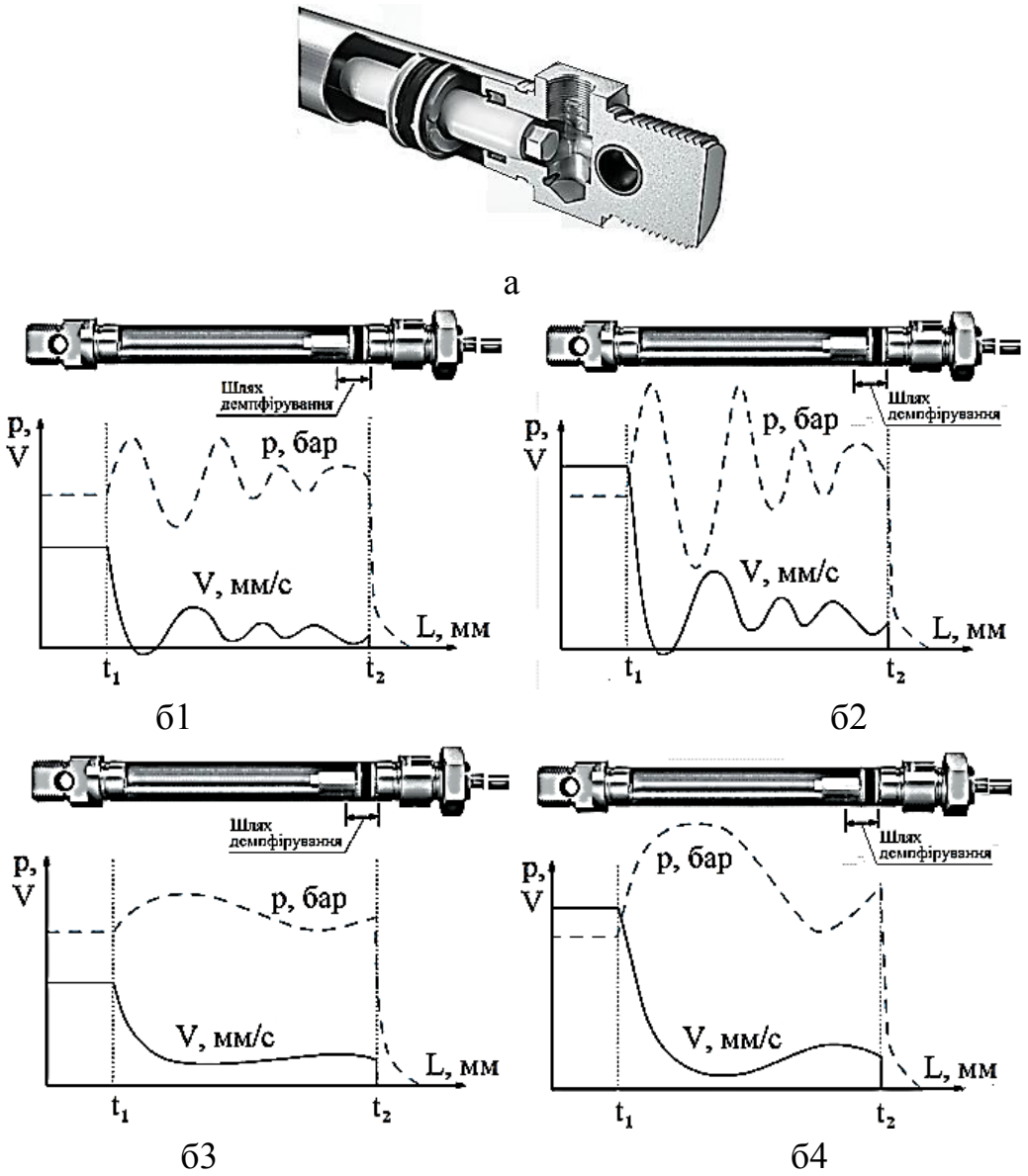
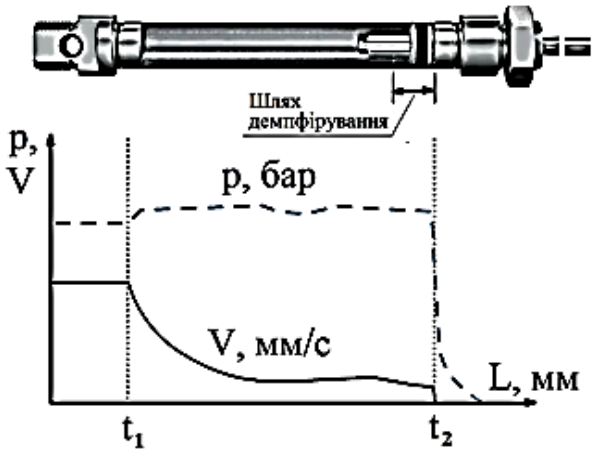
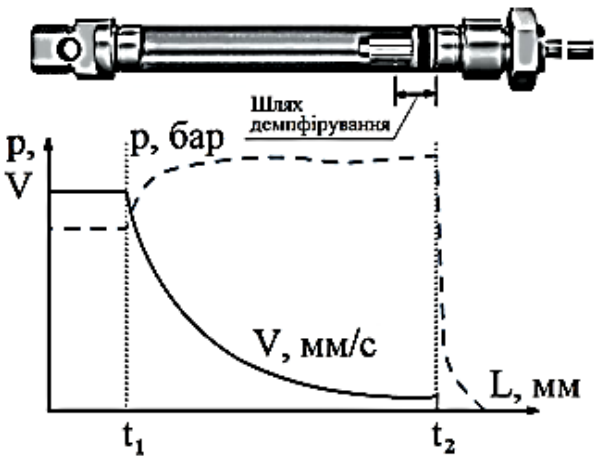


Рис. 6.2. Якісні характеристики поршня з подовженою довжиною демпфера (тип **PPV** демпфірування): а – 3D-схема конструкції; б – графіки динаміки роботи при малої (б1) і великої (б2) швидкості (без регулювання) та при малої (б3) і великої (б4) швидкості (з регулюванням)

а



б1



б2

Рис. 6.3. Якісні характеристики поршня з подовженою довжиною демпфера з використанням поздовжніх пазів у демпфіруючий частині поршня (тип **PPS** демпфірування): а - 3D-схема конструкції; б – графіки динаміки роботи при малої (б1) і великої (б2) швидкості

Поздовжні пази дозволяють повітрю виходити, що надає можливість динамічному і плавному переміщенню поршню в кінцеву позицію.

Замість звичайних пневматичних циліндрів в схемах мехатроніки використовують інноваційні біонічні м'язи (Fluidic Muscle MAS). «Біонічні м'язи» складаються з полого еластомерного циліндра, вбудованого в арамідні волокна. Коли такий м'яз наповнюється стислим повітрям, він збільшується в діаметрі і стискається по довжині, забезпечуючи пружний рух рідини.

Приклад застосування програмованих пневмоцилиндрів двосторонньої дії наведений на рис. 6.4, де прийняти наступні позначення:

1…4 – програмовані пневмоциліндри двосторонньої дії; 5 – портал;

6 – каретка; 7 – робочій інструмент; 8 – зубчасте-пасова передача;

A, B – штоки поршнів всіх циліндрів 1…4 у висунутому положенні;

\+ **A/B** – шток поршню циліндру 1 втягнутий , а шток поршню циліндру 3 у висунутому положенні;

А/+В – шток поршню циліндру 1 у висунутому положенні, а шток поршню циліндру 3 у втягнутому стані;

-А/В – шток поршню циліндру 2 у втягнутому стані, а шток поршню циліндру 3 у висунутому положенні;

А/-В – шток поршню циліндру 1 у висунутому положенні, а шток поршню циліндру 4 у втягнутому стані;

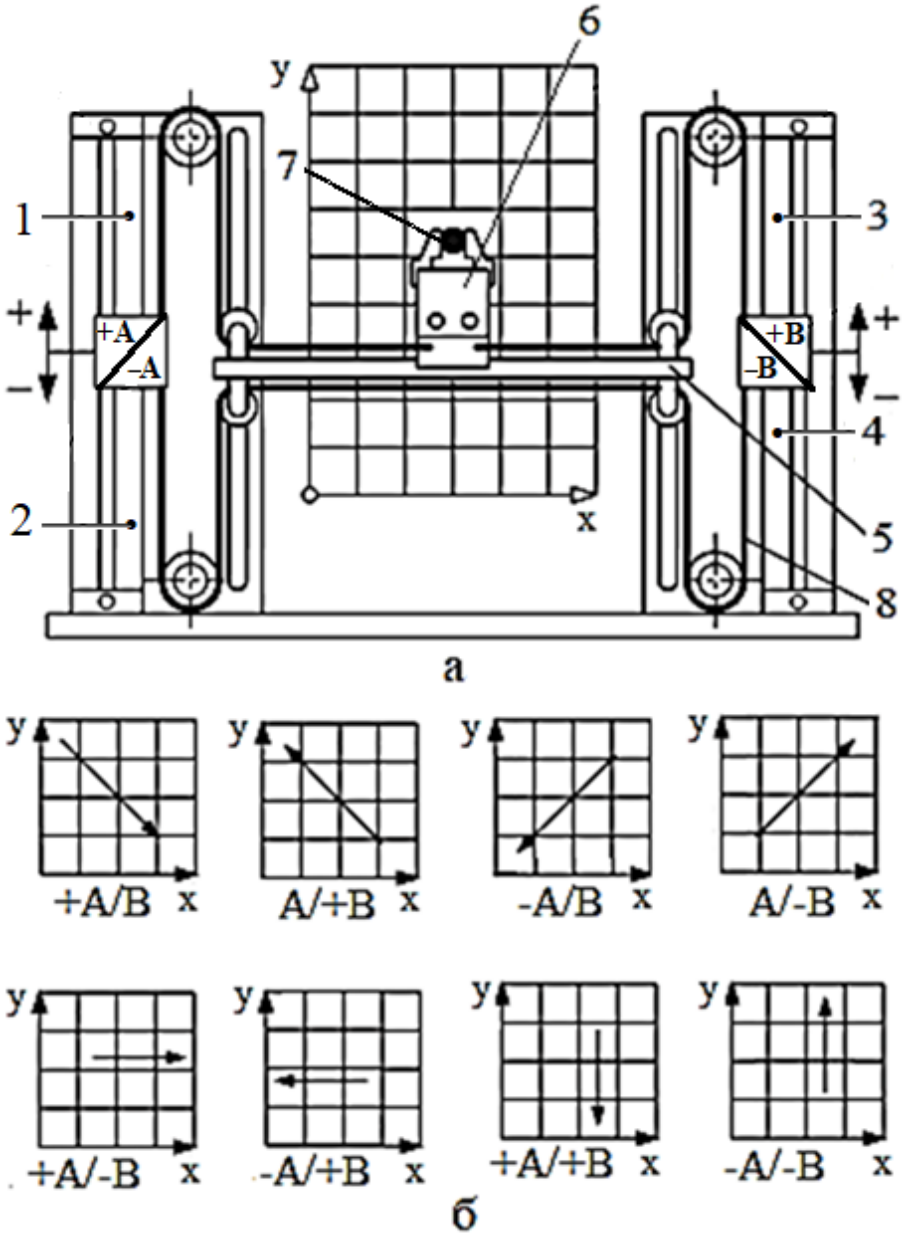


Рис. 6.4. Схема двокординатного механізму (а) із зубчасто-пасовою передачею для 8 положень (б) робочого інструменту 7, який закріплений на каретці 6 з програмно керованими пневмоприводами 1…4 двосторонній дії

+A/-B – шток поршню циліндру 1 втягнутий і шток поршню циліндру 4 у втягнутому стані;

-A/+B – шток поршню циліндру 2 втягнутий і шток циліндру 3 у втягнутому стані;

+A/+B – шток поршню циліндру 1 втягнутий і шток поршню циліндру 3 у втягнутому стані;

-A/-B – шток поршню циліндру 2 втягнутий і шток поршню циліндру 4 у втягнутому стані.

Наведені стани штоків з поршнями циліндрів відповідають наступному програмованого циклу роботи механізму на рис. 6.4:

$$\mathbf{1,2,3,4} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{1},\overline{\mathbf{3}} \rightarrow \overline{\mathbf{2}},\mathbf{3} \rightarrow \overline{\mathbf{4}} \rightarrow \overline{\mathbf{1}},\overline{\mathbf{4}} \rightarrow \overline{\mathbf{2}},\overline{\mathbf{3}} \rightarrow \overline{\mathbf{1}},\overline{\mathbf{3}} \rightarrow \overline{\mathbf{2}},\overline{\mathbf{4}} \qquad (6.1)$$

6.2. **Виконавчий механізм з програмованим пневмоприводом та вбудованою зубчасто-рейковою передачею**

Виконавчий механізм, у якому зворотно-поступовий рух поршню пневмоциліндру перетворюється в коливний рух веденої ланки має назву «*поворотний циліндр*», конструктивна схема якого наведена на рис. 6.6. Для такого перетворення руху в гільзу пнемоциліндру вбудована зубчаста-рейкова передача 4-5.

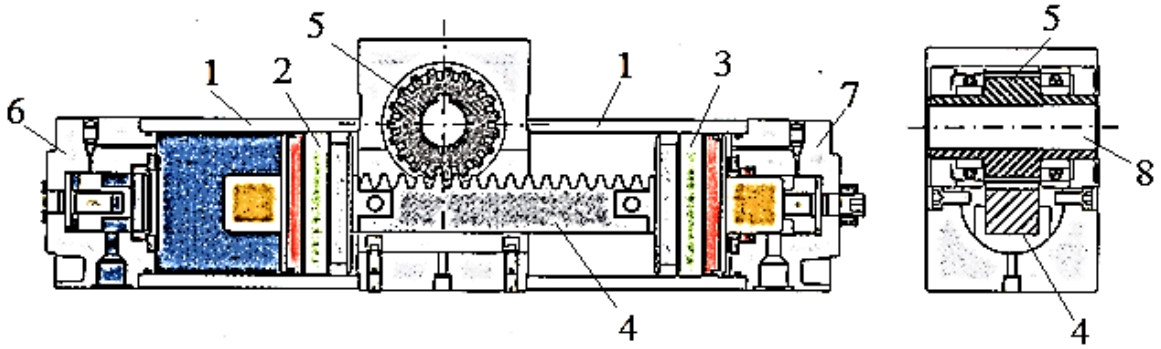


Рис. 6.5. Конструктивна схема поворотного пневмоциліндру: 1– гільза циліндру; 2– перший поршень з магнітним кільцем для визначення його положення; 3– другий поршень з магнітним кільцем; 4– зубчаста рейка; 5– зубчасте колесо; 6– ліва кришка з демпфером і вхідним каналом; 7– права кришка з демпфером і другім вхідним каналом; 8– вал

Зубчаста рейка 4 з'єднана з двома поршнями 2 і 3 і переміщується разом з ними в зворотно-поступовому напрямку в залежності від тиску у відповідної порожнини циліндру стислого повітря. Кришки 6 і 7 мають засоби демпфування поршнів в кінці їх переміщення. Кут повороту шестерні складає 90^0 або 180^0. Величина кута визначається довжиною зубчастої рейки, передаточним відношенням зубчастої передачі. Крутний момент

$$M = F \cdot r \text{ [Н·м]}, \tag{6.2}$$

де $F = p \cdot \pi \cdot d^2$ – поздовжня сила, яка створена тиском $\boldsymbol{p}$ стислого повітря на поршні циліндру діаметром d ;

r – радіус шестерні.

6.3. **Без штокові циліндри і тандем циліндри**

Без штокові циліндри. Якщо шток циліндру замінити на зовнішню каретку, яка кінематичне з'єднана з поршнем в гільзі циліндру, тоді отримуємо конструктивну модифікацію програмованого виконавчого механізму зворотно-поступової дії, який має поширену назву «*без штоковий пневмоциліндр*».

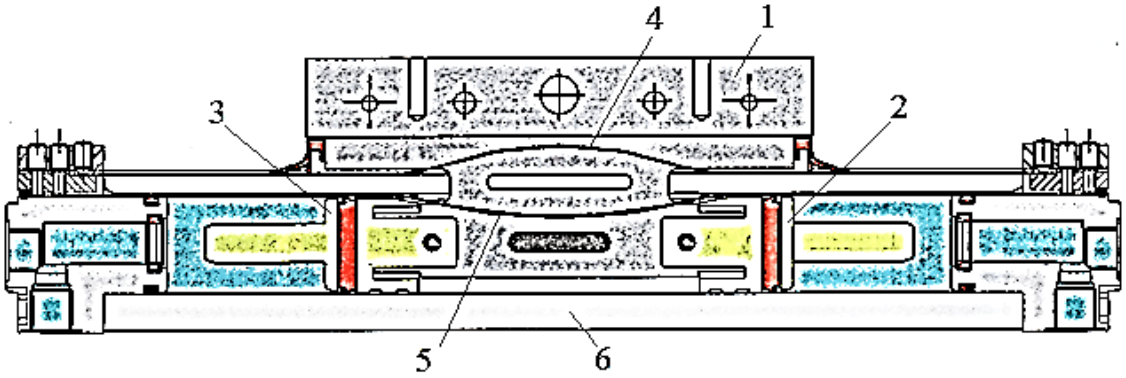


Рис. 6.6. Конструктивна схема без штокового циліндру з гальмуванням поршнів в кінці ходу

Для забезпечення значних лінійних переміщень без додаткових ланок передавальних механізмів фірмою Comozzi (Італія) розроблені пневматичні циліндри без штоку. На рис. 6.6 наведена конструктивна схема пневматичного циліндру без штока (без штоковий пневмоциліндр), у якому функцію штока виконує каретка 1. Каретка має кінематичний

зв’язок з поршнями 2 і 3 за допомогою тонких сталевих пластин 4 і 5. Пластини забезпечують герметизацію від зовнішній атмосфери поршнів, які переміщуються в гільзі 6. Полоса 4 закріплена на зовнішній поверхні гільзи, а полоса 5 – на внутрішній поверхні гільзи. При відсутності штоку площі поверхні поршнів однакові і тому при однаковому тиску в обох порожнинах циліндру зусилля, яке створює без штоковий циліндр для прямого руху і для реверсу також генеруються рівними по величіні. Але у без штокових циліндрах зовнішня сила і момент прикладаються ні до точці на осі руху поршнів, а зміщені до точки центру маси на осі каретки, тому обмежуються величина навантажень на каретку з боку об’єктів циклового керування в мехатронних системах.

Тандем циліндри. Для збільшення зусилля на штоці циліндру два або більше циліндрів з’єднують послідовно в один ряд в одному корпусі. Значення результуючого зусилля F визначається за виразом:

$$F = p \cdot \sum_{i=1}^{N} S_i \,, \tag{6.3}$$

де $S_i = \pi d^2$ – площа поршнів у безштокової порожнині;

p – тиск пневмомагістралі;

N – кількість циліндрів.

Якщо N = 2 отримуємо зусилля для тандем-циліндру (рис. 6.7,а), при N=3 для трайдема (рис. 6.7, б), при N=4 для тетрадема. Інша назва таких циліндрів двоступінчастий, триступінчатий, чотириступінчатий і т. д. Обмеження в кількості циліндрів накладається обмеженням габаритних розмірів програмованого приводу.

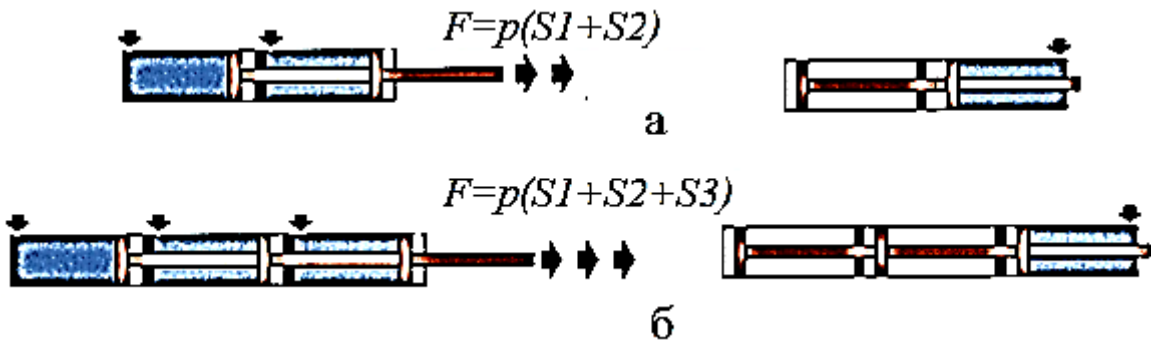


Рис. 6.7. Принципові схеми роботи тандем-циліндру (а) і три-циліндру (б)

6.4. **Опозитні циліндри і мультіпозиційні циліндри**

Опозитні циліндри. Якщо два циліндра двосторонній дії з однаковою або різною величиною переміщенням поршнів з'єднати між собою в стик кришками, тоді можна отримати три або чотири позиції штока на виході, як це зображено на рис. 6.8. Такі циліндри мають назву «*опозитні циліндри*».

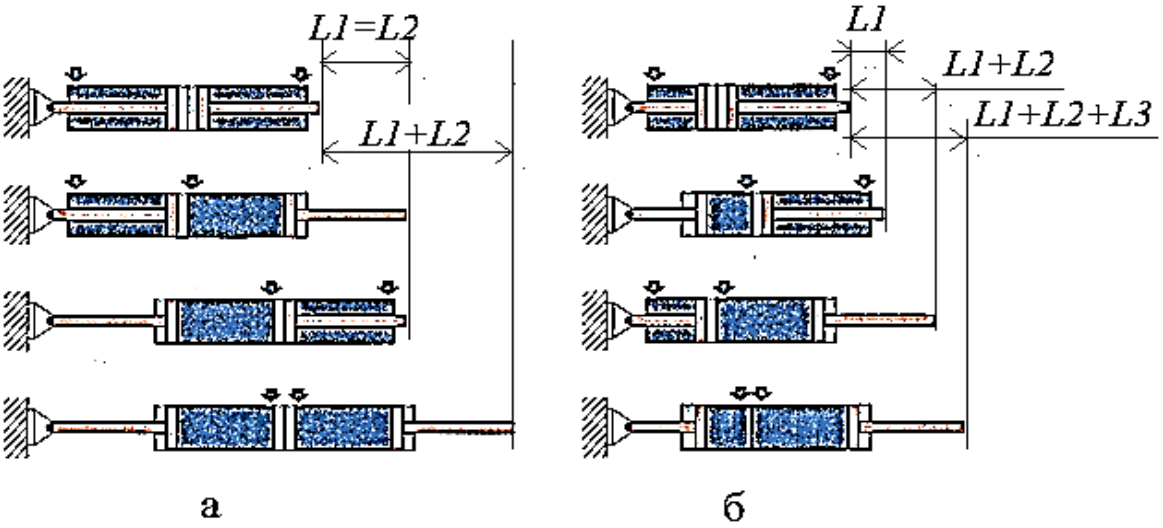


Рис. 6.8. Принципові схеми роботи опозитних циліндрів: а – три позиції штоку, якщо стикуються циліндри з однаковим ходом поршнів (*L1=L2*); б – чотири позиції штоку, якщо стикуються циліндри з різним ходом поршнів ($L1 \neq L2$)

Мультіпозиційні циліндри. Якщо в одної гільзі циліндру розташовано декілька послідовно розташованих пар «*поршень-шток*», які механічно не з'єднані один з одним, тоді отримаємо декілька позицій зупинення. При подачі тиску в перший циліндр його шток штовхає поршень другого циліндру (рис. 6.9,б). При подачі тиску в другий циліндр його шток переміщується на відстань $L = 2\text{x}$ (рис. 6.9,в).

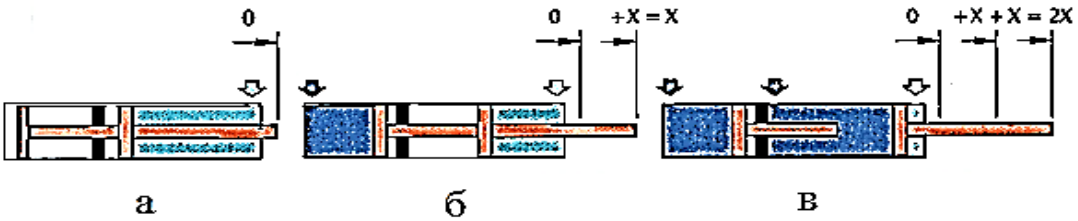


Рис. 6.9. Принципові схеми роботи мультіпозиційних циліндрів: а – вихідне положення; б – $L = \text{x}$; в – $L = 2\text{x}$

Якщо конструктивне з’єднати три циліндра з різними ходами штоків за схемою на рис. 6.9, то отримуємо $2^3 = 8$ різних позицій штоку третього циліндру, тобто 1 байт кодованих значень ходів виконавчого механізму мехатронного модуля.

6.5. **Комбіновані циліндри**

Для реалізації циклу руху за законом «швидко-повільно» мехатронний модуль має два виконавчих механізму зворотно-поступової дії у вигляді пневмоциліндру і гідроциліндру, штоки яких розташовані паралельно (рис. 6.10).

Наприклад, підведення свердла для свердлення на свердлильному верстаті-автоматі повинно бути швидким (холостий хід), а робоча подача свердла на потрібну глибину повинна виконуватися при повільному і рівномірному русі свердла. Тому спочатку на ділянці холостого ходу *L1* працює пневмоциліндр до моменту утворення тиску у без штокової порожнині циліндру тиску, який зрівнюється з тиском нестислої рідини в гідроциліндрі. На ділянці робочого ходу *L2* штоків пневмоцилідру і гідроциліндру рухаються разом (рис. 6.10,в).

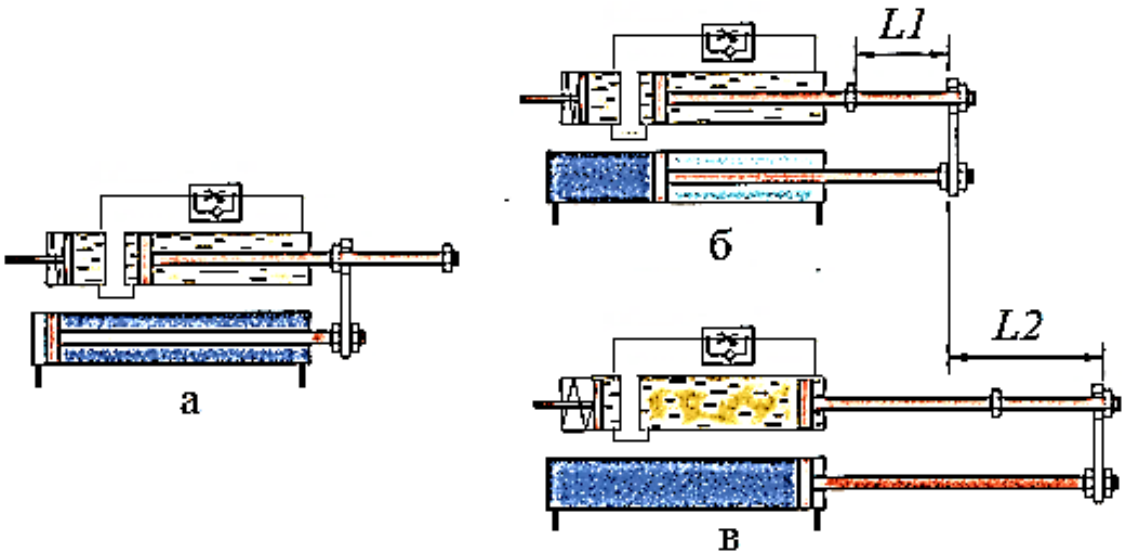


Рис. 6.10. Принципові схеми роботи комбінованих циліндрів: а – вихідне втягнуте положення; б – холостий хід $L1$ (швидкий рух поршня пневмоциліндру); в – робочий хід 2 (повільний рух поршня гідравлічного гальма)

6.6. **Двопозиційний БІстабільний пневморозподільник з електромагнітним керуванням**

Пневморозподільник призначений для запуску, зміни напрямку і зупинки потоку стисненого повітря в двох або більше зовнішніх пневмолініях під впливом керуючого пристрою. Під зовнішніми пневмолініями розуміють канали для проходження повітря і повітропроводи, у тому числі для зв'язку з атмосферою. По способу керування пневморозподільники поділяються на розподільники з механічним, пневматичним, електромагнітним чи комбінованим керуванням. За можливістю зберігання положення після зняття керуючого сигналу: БІстабільні та МОНОстабільні. За типом запірного елементу: золотникові, клапанні та кранові. По кількості фіксованих положень: двопозиційні та трьох позиційні. За кількістю ліній підводу та відводу повітря дволінійні, трьохлінійні та багатолінійні.

Принципова конструкція у вигляді конструктивної схеми типового БІстабільного пневморозподільника приведена на рис. 6.11. *Золотник* 1 переміщується в гільзі 2 запресованій в корпусі 3. *З о л о т н и к* це елемент у вигляді металевого циліндру, який має одну проточку у розподільниках типу 3/2 і дві проточки у розподільниках типу 5/3.

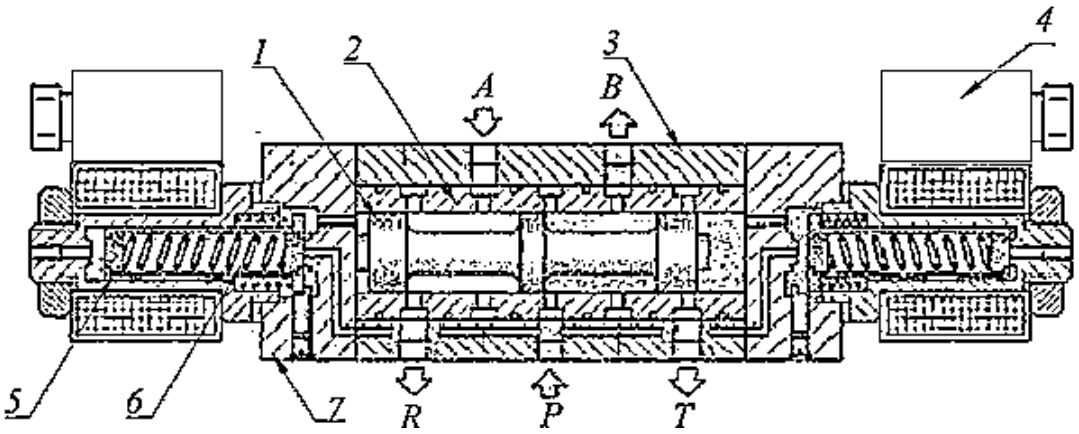


Рис. 6.11. Конструктивна схема БІстабільного пневморозподільника

При подачі керуючого сигналу на лівий або правий електромагніт 4. Якір електромагніту переміщує штовхач з клапаном 5 навантажений пружиною 6, з’єднуючи лінію нагнітання **P** через відкритий отвір у кришки 7 пневморозподільника з торцем золотника 1. В результаті золотник зміщується в одне з крайніх положень, з’єднуючи лінію

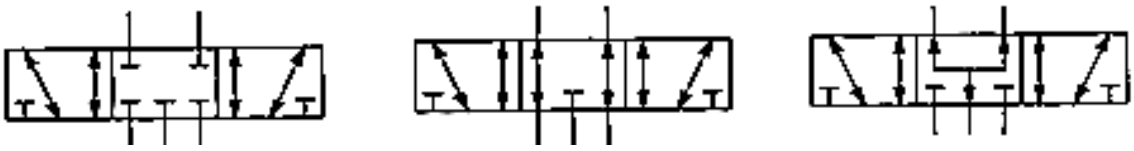
нагнітання **P** з відповідною виконавчою лінією **A** чи **B**, а іншу виконавчу лінію з відповідним каналом вихлопу в атмосферу **R** чи **T**. При знятті сигналу золотник залишається в останньому положенні. У випадку з МОНОстабільним керуванням, замість одного з електромагнітів 4 на торець золотника давить пружина.

Пневморозподільник виступає як підсилювач по потужності, оскільки потужність сигналу його перемикання на декілька порядків нижча за потужність яку отримує пневмоциліндр. Тобто можна вважати, що пневморозподільник виконує функцію імпульсного підсилювача в пневматичної системі. Таким імпульсним підсилювачем електричного сигналу в електричних і електронних схемах є напівпровідниковий транзистор.

При переході до алгоритмічної мови STL, для програмування контролера потрібно сигнали Y_i стану (включено/вимкнено) команди керування пневморозподільником адресувати виходам O_i («***O**utput*») контролера, а сигнали контролю стану положення штока пневмоциліндра за допомогою кінцевих вимикачів адресувати входам I_i («***I**nput*») контролера.

6.7. **Трипозиційні пневморозподільники**

Для позиціювання виконавчих механізмів зворотно-поступової дії в трьох положеннях в схемах мехатроніки використовують пневморозподільники з трьома позиціями (рис. 6.12):



а б в

Рис. 6.12. Умовні позначення пневморозподільників п’яти лінійних трипозиційних (тип 5/3):

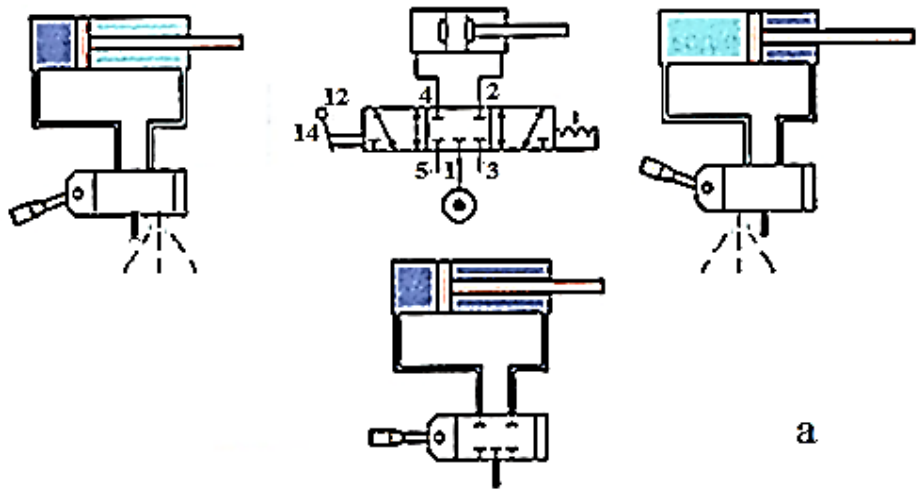
а – із ***закритим центром*** (внутрішні порожнини виконавчого пневмоциліндру у середньому (третьому) положенні знаходяться під тиском і не з’єднані з магістраллю живлення);

б – з ***відкритим центром*** (внутрішні порожнини виконавчого пневмоциліндру у середньому (третьому) положенні знаходяться під атмосферним тиском);

в – з ***центром під тиском*** (внутрішні порожнини виконавчого пневмоциліндру у середньому (третьому) положенні знаходяться під тиском і з'єднані з магістраллю живлення). Схеми підключення пари «3-позиційний пневмоциліндр – пневморозподільник типу 5/3» наведені на рис. 6.13.

Трипозиційні пневморозподільники з електропневматичним і / або пневматичним керуванням є МОНОстабільними, тобто при відсутності управляючих сигналів розподільник повертається у третю (центральну позицію) під дією стиснутої зворотній пружини. МОНО- і БІстабільні розподільники з ручним керуванням управляються оператором.

В схемі «*із закритим центром*» (рис. 6.13,а) при крайньому лівому положенні штока вихлоп 3 пневморозподільника з'єднує штокову порожнину циліндру з атмосферним тиском, при крайньому правому положенні штока вихлоп 5 пневморозподільника з'єднує без штокову порожнину циліндру з атмосферним тиском, а при середньому положенні штока вихлопи 3 і 5 пневморозподільника відсікають циліндр від атмосферного тиску і поршень залишається заблокованим тиском повітря, що залишилося в обох порожнинах циліндру. В аварійної ситуації поршень зі штоком не можна зсунути вручну з центральної позиції.



а

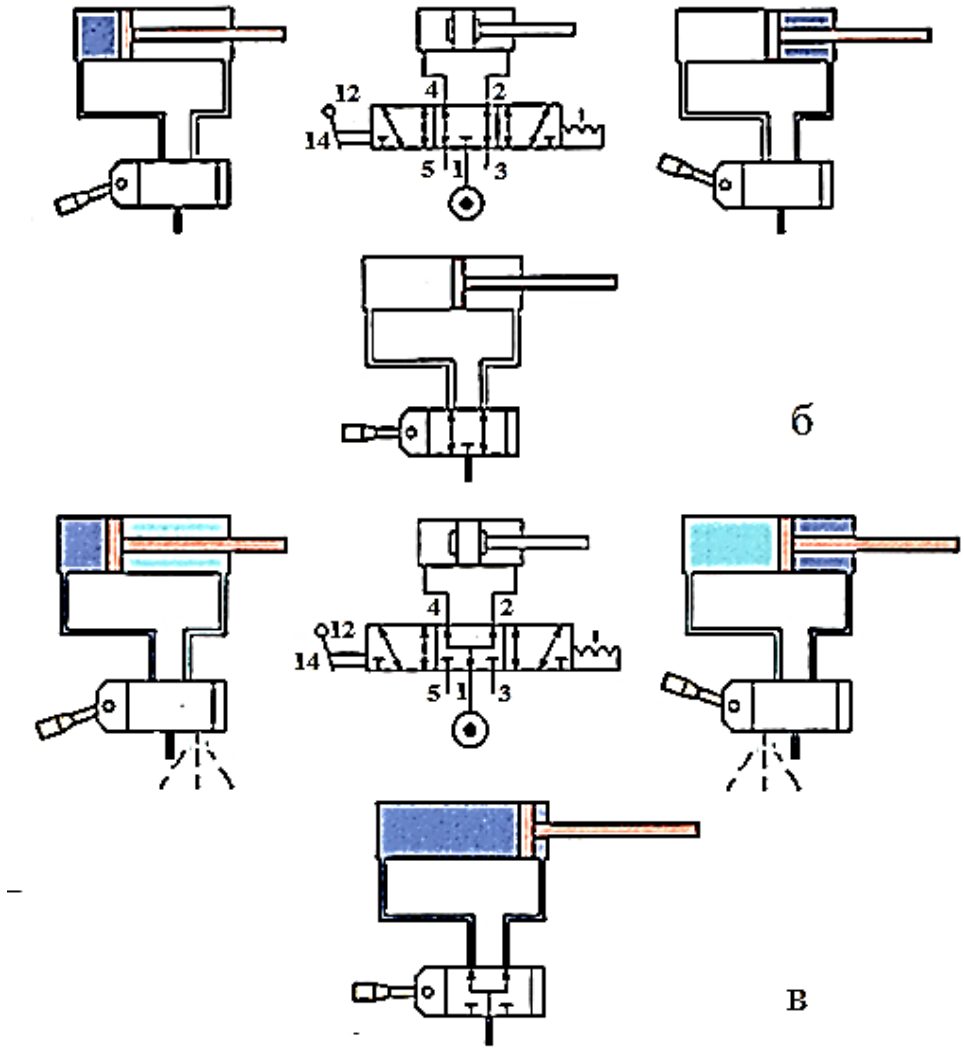


Рис. 6.13. Схеми ручного керування трипозиційними пневмо розподільниками: а – модифікація 1 «із закритим центром»; б – модифікація 2 «з відкритим центром»; в – модифікація 3 «з центром під тиском»

В схемі «*з відкритим центром*» (рис. 6.13,б) в центральної позиції штока вхід 1 закритий для стислого повітря, а виходи 2 і 4, які з'єднані з порожнинами циліндру відкрити в сторону отворів вихлопу 3 і 5. І тому в штокової і у без штокової порожнинах циліндру тиск атмосферний. Шток з поршнем можна вільно рухати вручну при потребі. Такий розподільник використовують для забезпечення безпеки при зникненні управляючих сигналів, тобто в такої ситуації відбувається скидання стислого повітря в в порожнинах циліндру і привод стає "знесиленим".

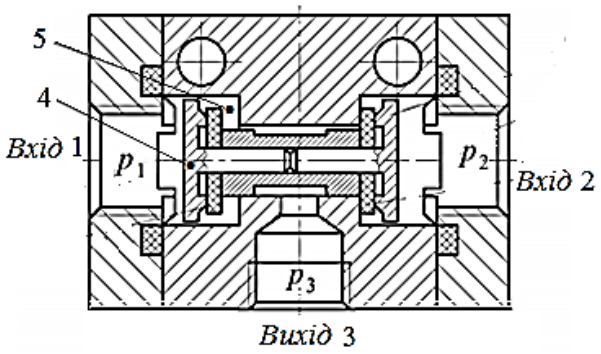
В схемі «*з центром під тиском*» (рис. 6.13,в) в центральної позиції штока обидві порожнини циліндра знаходяться під тиском. Переміщення

штоку і поршня в крайнє ліве положення відбувається при утворенні каналу 1-2, а в крайнє праве положення при утворенні каналу 1-4 для стислого повітря. В аварійної ситуації поршень зі штоком можна зсунути вручну з центральної позиції.

6.8. **Логічні пневмоклапани**

При створенні пневматичних схем мехатроніки використовують пневматичні логічні елементи, що виконують логічні операції «І» (рис. 6.14) та «АБО» (рис. 6.15). На рис. 6.15,а показана конструкція пневмоклапана «І», а на рис. 6.15,б - таблиця логічних сигналів.

Якщо, тиск *p2 на вході 2* перевищить тиск *p1 на вході 1* штовхач 4 переміщується вліво і за рахунок ущільнення перекриває вікно 5. Надходження стислого повітря на *вихід 3* припиняється і на вихід 3 буде надходити стисле повітря *зі входу 1* під тиском *p1* (перша стрічка таблиці). У випадку, якщо тиск *p1 на вході 1* перевищить тиск *p2 на вході 2* штовхач 4 переміщується вправо і за рахунок ущільнення перекриває вікно 5. Надходження стислого повітря на *вихід 3* припиняється і на вихід 3 буде надходити стисле повітря *зі входу 2* під тиском *p2* (друга стрічка таблиці).



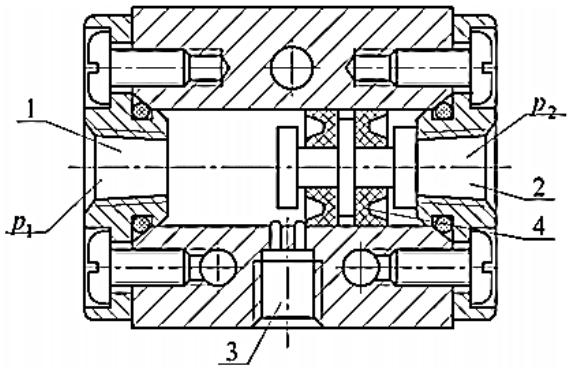
p1	*p2*	*p3*
0	**1**	**0**
1	**0**	**0**
0	**0**	**0**
1	**1**	**1**

а б

Рис. 6.14. Конструкція пневмоклапана «І» (а) і таблиця логічних сигналів (тиску) на входах *p1* , *p2* і виході *p3* при виконанні логічної операції «кон'юнкція» (*p1 AND p2*)

На виході 3 сигнал (тиск) з'являється тоді, на вході 1 і на вході 2 буде однаковий надлишковий тиск робочого середовища (*p1=p2*) - четверта стрічка таблиці.

На рис. 6.15 показана конструкція пневмоклапана «АБО» і таблиця логічних сигналів.



а

p1	*p2*	*p3*
1	**0**	**1**
0	**1**	**1**
1	**1**	**1**
0	**0**	**0**

б

Рис. 6.15. Конструкція пневмоклапана «АБО» (а) і таблиця (б) логічних сигналів (тиску) на входах *p1*, *p2* і виході *p3* при виконанні логічної операції «диз'юнкція» (*p1 OR p2*)

При наявності тиску *p1* (*p1* = 1) в камері 1 і відсутності тиску *p2* в камері 2 (*p2* = 0) плунжер 4 переміщується вправо і повітря під тиском *p1* надходить в камеру 3 (*p3*=1). Якщо тиск *p2* = 1 і тиск *p1*=0, тоді плунжер 4 переміщується вліво і повітря під тиском *p2* надходить в камеру 3 (*p3*=1).

6.9. **Типові механізми з програмованим пневмоприводом мехатронних систем**

Кулачкові механізмі з програмованим пневмоприводом

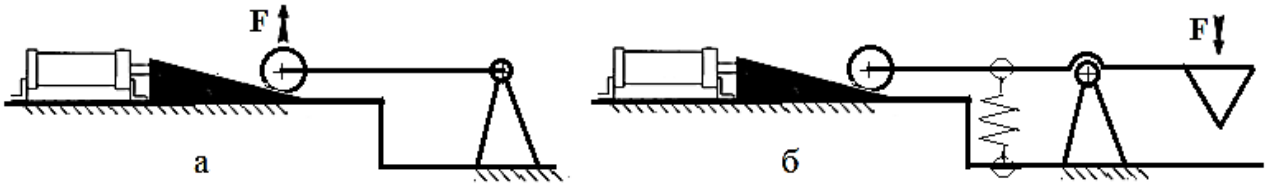
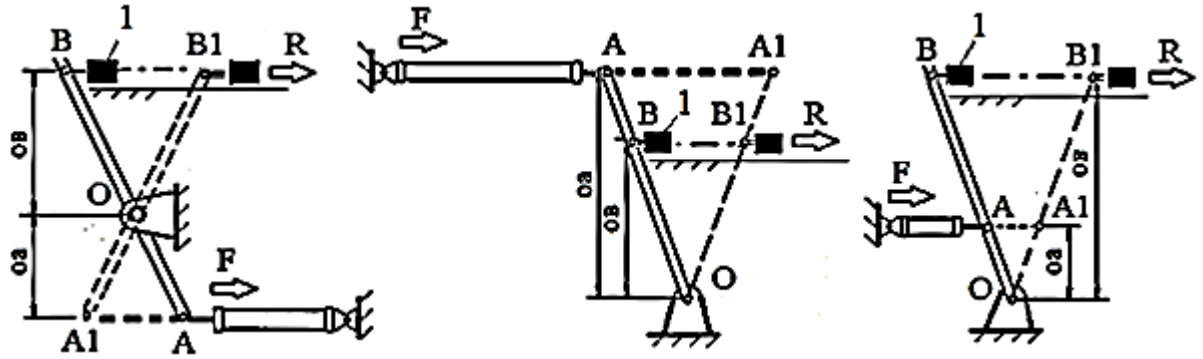


Рис. 6.16. Кінематичні схеми кулачкових механізмів з продовжним рухом кулачка-клину: а – коромисловий механізм з кінематичним замиканням пари "кулачок-штовхач; б – важільний з силовим замиканням пари "кулачок-штовхач

Важільні механізми з програмованим пневмоприводом

Модулі програмованого переміщення робочого інструменту по горизонталі застосовується у CNC-верстатах (верстатах з числовим програмним керуванням) для зміни інструменту, у маніпуляторах при переміщенні вантажів на автоматизованих складах, при автоматизованому поштучному відокремленні плоских деталей зі стосу знизу у магазинних завантажувальних пристроях, встановлення заготовки і зняття обробленої деталі та в інших випадках при застосуванні завантажувально-розвантажувальних мехатронних модулів з БІстабільним або МОНОстабільним керуванням.

На рис. 6.17 наведені кінематичні схеми типових мехатронних модулів з передаточними важільними механізмами 3х модифікацій з програмованим пневмоприводом, у яких застосовані передаточні механізми з різним родом важелів по визначенню теорії механізмів і машин. При цьому утворюються різні передаточні функції, які змінюють робочі характеристики пневмоприводу і геометричні параметри структурної схеми механізму.



а б в

Рис. 6.17. Розрахункові схеми коромислових механізмів з програмованим пневмоприводом: а – важіль першого роду; б – важіль другого роду; в – важіль третього роду

Для мехатронних модулів на рис. 6.17 розглянемо приклади розрахунку функціонально-адекватних передаточних механізмів різних структур з програмованим пневмоприводом для поштучному відокремленні плоских деталей низу взуття зі стосу знизу у магазинних завантажувальних пристроях для автоматизації взуттєвих машин.

Завдання: розрахувати силу F пневмоциліндру для переміщення деталей низу взуття на відстань $BB_1 = 55$ мм при їх поштучному відокремленні зі стосу, якщо у всіх випадках реакція R від сил опору при відокремленні нижньої деталі зі стосу складає R = 12 Н.

Приклад 1 (рис. 6.17,а). Точка О стояку двоповодкового коромисла АВ знаходиться між точкою А кріплення штоку поршня пневмоциліндру і точкою В кріплення штовхача 1 (*важіль першого роду*).

Дано: довжина АВ = 110 мм; ОВ = 66 мм; деталь потрібно переміщувати на відстань BB_1 = 55 мм за допомогою штовхача 1 для наступного передавання у валки машини (на схемі не показана). Відрізки AA_1 і BB_1 паралельні.

Визначити: хід AA_1 поршню зі штоком і значення сили F.

Визначаємо відстань від точку опори О до точку прикладання рушійної сили:

$$OA = AB - OB = 110 - 66 = 44 \text{ мм}.$$

З подібних трикутників AOA_1 і BOB_1 можна записати:

$$\frac{OA}{OB} = \frac{AA_1}{BB_1} \tag{6.4}$$

З виразу (1) визначаємо хід AA_1 поршню зі штоком:

$$AA_1 = \frac{OA \cdot BB_1}{OB} = \frac{44 \cdot 55}{66} = 36{,}6 \text{ мм} \tag{6.5}$$

Довжина плеч **оа** і **ов** дорівнюється:

$$\text{оа} = \sqrt{(OA)^2 - (0{,}5 \cdot AA_1)^2} = \sqrt{(44)^2 - (0{,}5 \cdot 36{,}6)^2} \cong 40 \text{ мм} \tag{6.6}$$

$$\text{ов} = \sqrt{(OB)^2 - (0{,}5 \cdot BB_1)^2} = \sqrt{(66)^2 - (0{,}5 \cdot 55)^2} \cong 60 \text{мм} \tag{6.7}$$

З умови рівності моментів

$$F \cdot \text{оа} = R \cdot \text{ов} \tag{6.8}$$

З виразу (6.9) визначаємо силу F пневмоциліндру, яка потрібна для переміщення деталей низу взуття на відстань $BB_1 = 55$ мм при їх поштучному відокремленні зі стосу для реакція R = 12 Н від сил опору :

$$F = \frac{R \cdot \text{ов}}{\text{оа}} = \frac{12[\text{Н}] \cdot 0.06[\text{м}]}{0.04[\text{м}]} = 18 \text{ Н} \qquad (6.9)$$

Приклад 2 (рис. 6.17,б). Точка О стояку коромисла АВ з і точка А прикладання рушійної сила F пневмоциліндру знаходиться на кінцях коромисла АО, а реакція R= 12 Н від сил опору прикладена в точці В коромисла (*важіль другого роду*).

Дано: довжина ОА = 110 мм; ОВ = 66 мм; деталь потрібно переміщувати на відстань BB_1 = 55 мм за допомогою штовхача 1 для наступного передавання у валки машини. OB_1= 66 мм. Плечо оа = 100 мм.

Визначити: хід AA_1 поршню зі штоком і значення сили F.

З подібних трикутників AOA_1 і BOB_1 та виразу (6.4) визначаємо хід AA_1 поршню зі штоком :

$$AA_1 = \frac{OA \cdot BB_1}{OB} = \frac{110 \cdot 55}{66} = 91{,}6 \text{ мм} \qquad (6.10)$$

З урахуванням довжини плеча **оа** за виразом (6.6) і довжини плеча **ов** за виразом (6.7) та умови рівності моментів з виразу (6.8) визначаємо силу F пневмоциліндру, яка потрібна для переміщення деталей низу взуття на відстань $BB_1 = 55$ мм при їх поштучному відокремленні зі стосу для реакція R = 12 Н від сил опору при відокремленні нижньої деталі зі стосу:

$$F = \frac{R \cdot \text{ов}}{\text{оа}} = \frac{12[\text{Н}] \cdot 0.06[\text{м}]}{0.10[\text{м}]} = 7.2 \text{ Н} \qquad (6.11)$$

Приклад 3 (рис. 6.17,в). Рушійна сила F пневмоциліндру прикладена між точкою О стояку і точкою В реакції R= 12 Н від сили опору (*важіль третього роду*).

Дано: довжина ОВ = 110 мм; при поштучному відокремленні деталі зі стосу нижню деталь потрібно переміщувати на відстань BB_1 = 55 мм за

допомогою штовхача 1 для наступного передавання у валки машини. Плечі ов = 100 мм і оа = 40 мм.

Визначити: хід AA_1 поршню і значення рушійної сили F пневмоциліндру для переміщення деталей на відстань B_1B.

З подібних трикутників AOA_1 і BOB_1 та виразу

$$\frac{\text{оа}}{\text{ов}} = \frac{AA_1}{BB_1} \tag{6.12}$$

визначаємо хід AA_1 поршню зі штоком :

$$, \qquad AA_1 = \frac{\text{оа}\cdot BB_1}{\text{ов}} = \frac{40\cdot 55}{100} = 22 \text{ мм} \tag{6.13}$$

З урахуванням рівності моментів F·оа = R·ов визначаємо силу F необхідну для утримання вантажу масою m = 1,2 кг:

$$F = \frac{R\cdot \text{ов}}{\text{оа}} = \frac{12[\text{Н}]\cdot 0.10[\text{м}]}{0.04[\text{м}]} = 30 \text{ Н} \tag{6.14}$$

Таким чином, згідно отриманих різних значень для сил за виразами (6.9), (6.11) і (6.14) структура передаточного механізму з програмованим пневмоприводом визначає різні значення рушійної сили F для обрання різних програмованих пневмоциліндрів механізму переміщення деталей на однакову відстань B_1B = 55 мм при однакової сили опору з боку деталей, які поштучно відокремлюються зі стосу магазинного завантажувального пристрою.

Для механізмів на рис. 6.17 цикл 1-N1 може бути реалізований з використанням програмованого контролера (розділ 9) або без використання програмованого контролера (розділ 8).

Другий приклад використання важільного механізму за схемою на рис. 6.17,а) наведений на рис. 6.18 у конструкції мобільного прес-фітингу з програмованим пневмоприводом, при запресуванні підшипників у корпус стаціонарної заготівлі.

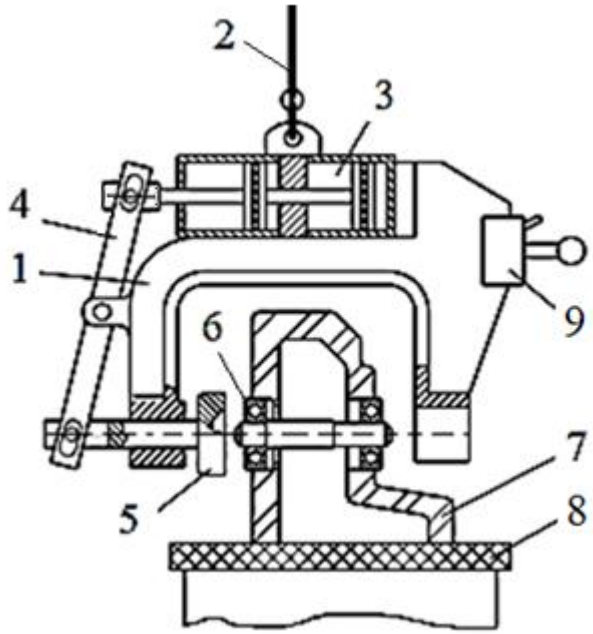


Рис. 6.18. Принципова схема конструкції мобільного пнемо- пресу за схемою важільного механізму на рис. 6.17,а: 1 – корпус преса-кронштейна; 2 – трос; 3 – тандем-циліндр (рис. 6.7,а); 4 – важіль (двоплече коромисло); 5 – пресуюча ланка; 6–підшипник, який запресовується в корпус; 7 – основний компонент збірки стаціонарної заготівлі; 8 – опора стаціонарній заготівлі; 9 – елемент керування.

Монтажні роботи, наприклад, при запресуванні підшипників на великих заготовок часто виконуються із стаціонарними заготівлею (корпусом) на значної відстані від місця збірки машини. Це стосується і ремонтних операцій. Тому прес для зборки має прийти до заготовок і відповідно бути рухливим і підвішений з підйомника. Виконавчий механізм з пневмоприводом вмонтований у корпус прес-кронштейну і застосовує силу до веденої (пресуючої) ланки через коромисло 5, яке є важелем першого роду. Сила діє по замкнутої петлі через кронштейн на протилежну опору. Операції з демонтажу також можуть бути виконані з цим мобільним пристроєм, якщо підходять відповідні інструменти.

Конструктивні особливості нерухомих і рухомих варіантів кріплень циліндру до станіни у важільних механізмах наведені на рис. 6.19 і рис. 6.20.

При нерухомому кріплення циліндру до станіни (прис. 6.19) поршень і шток є повзуном з одним ступеням рухомості і застосовується у кулачкових (рис. 6.16), повзунно-коромислових (рис. 6.18) або повзунно-

кривошипних програмно керованих механізмах машин. В цих випадках поршень і шток рухаються тільки зворотно-поступово по осі координат в залежності від площини закріплення циліндру.

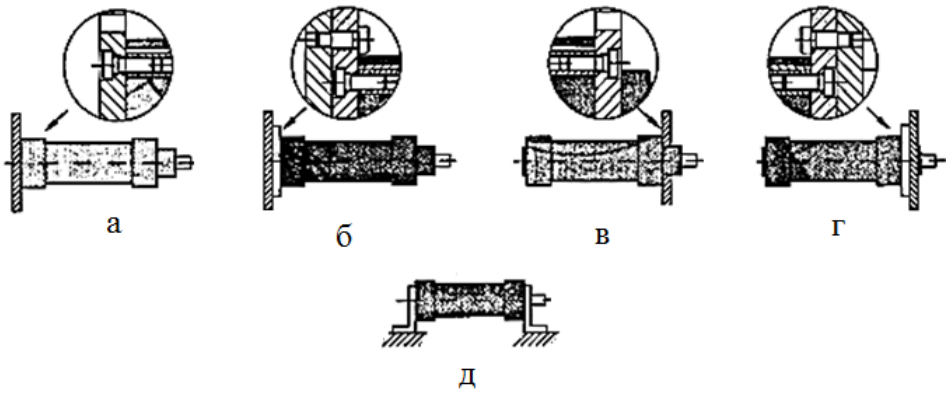


Рис. 6.19. Варіанти нерухомого кріплення циліндрів до станіни: а, б – лівостороннє; в, г – правостороннє; д – двостороннє

При рухомому варіанті кріплення циліндру до станіни (прис. 6.20) поршень і шток утворюють кулісу з двома ступенями рухомості тому, що рухаються зворотно-поступово відносно циліндру і одночасно повертаються в площині разом з циліндром відносно осі кріплення циліндру.

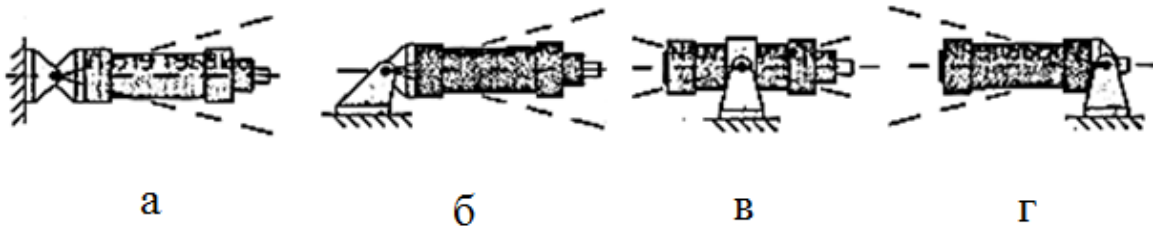


Рис. 6.20. Варіанти рухомого кріплення циліндрів до станіни: а, б – лівостороннє; в – центральне; г – правостороннє

При переміщенні веденої ланки по горизонтальної поверхні, яка розташована строго паралельно осі циліндра можливо пряме різьбове з’єднання штоку з ланкою (рис. 6.21,а).

При неточності монтажу виникають радіальні навантаження на шток, втулку і манжети циліндра. Для компенсації неточності монтажу з’єднання штоку з веденою ланкою передавальних механізмів виконується за допомогою рухомих кінематичних пар з одної (рис. 6.21,б), двома (рис. 6.21,в) або з трьома (рис. 6.21,г) ступенями вільності руху.

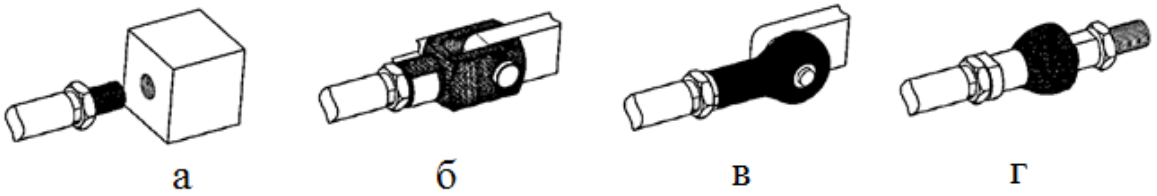


Рис. 6.21. Варіанти з'єднань штоку поршню з веденою ланкою передавальних механізмів: а – різьбове; б – вільчасте; в – сферичною кінематичною парою; г –кульовим шарніром

Приклад застосуванні кулісного механізму з програмованим пневмоприводом та вбудованою зубчасто-рейковою передачею або пневмоприводу обертової дії наведений на рис. 6.22,а. В кулісному передавальному механізмі для поштучного відокремлення і перевантаження деталей з позиції 8 на позицію 7 куліса 3 отримає рух від коромисла 2 (рис. 6.22,б і рис. 6.22,в). Функціональна схема мехатроніки за вимогами стандарту **VDI 2860** (розділ1, таблиця 1.2) наведена на рис. 6.22,г.

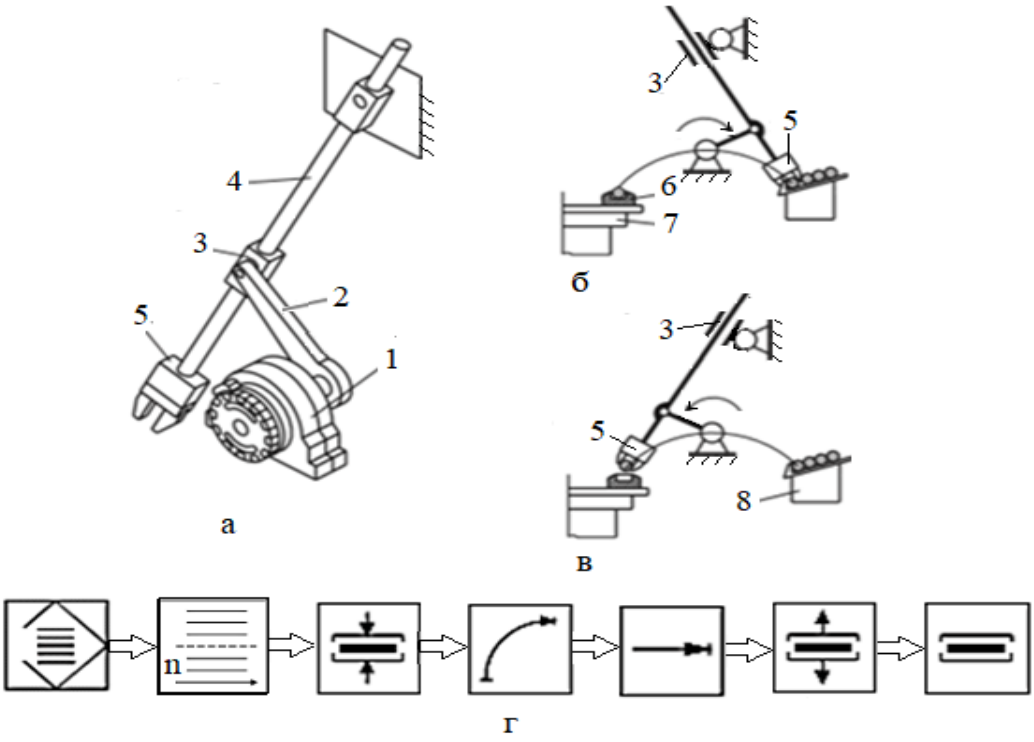


Рис. 6.22. Кулісний передавальний механізм з програмованим пневмоприводом: а – конструктивне-кінематична 3D-схема; б – кінематична 2D-схема (позиція розвантаження); в – кінематична 2D-схема (позиція завантаження); г – на функціональна схема мехатроніки за вимогами стандарту **VDI 2860** (таблиця 1.2)

На рис. 6.22 прийняти наступні позначення: 1 – програмований пневмопривод обертової дії або виконавчий механізм з програмованим пневмоприводом та вбудованою зубчасто-рейковою передачею (рис. 6.5); 2 – коромисло; 3 – куліса; 4 – важіль куліси; 5 – захват деталі (виробу); 6 – приймач деталі (виробу); 7 – місце завантажування деталей; 8 – місце розвантажування деталей (виробів).

У двокординатного механізмі із зубчасто-пасовою передачею (рис. 6.4) замість чотирьох програмованих пневмоприводів двосторонній дії може бути використані два двокоромислових механізмів з програмованими пневмоприводами обертової дії або два сервопривода з програмованим кутом повороту ведучої ланки (коромисла), як це зображено на рис. 6.23.

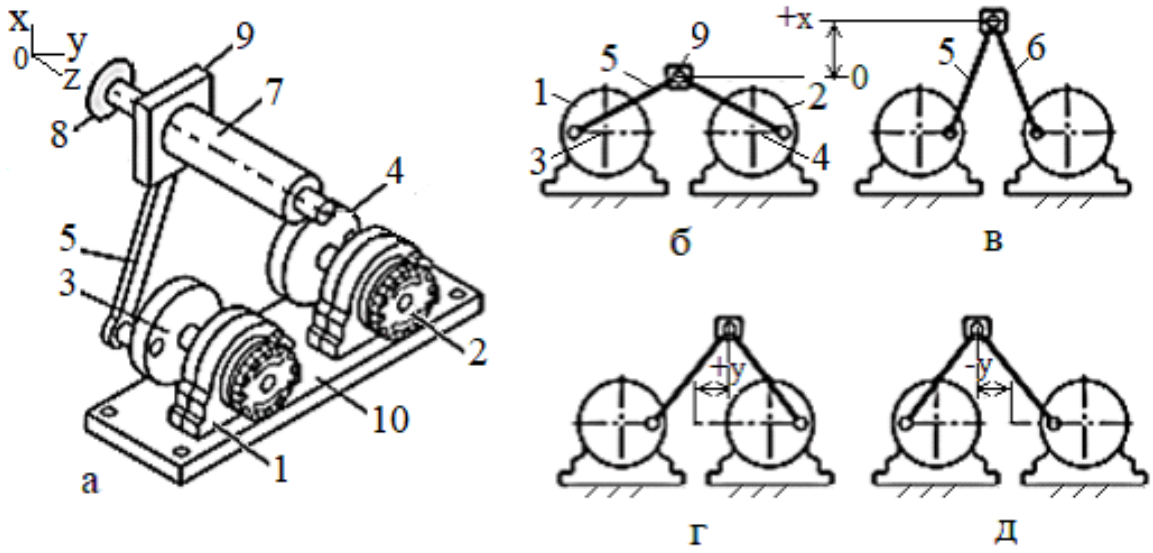


Рис. 6.23. Двокоромисловий чотири позиційний 3D-механізм з програмованими пневмоприводами: а – конструктивне-кінематична схема; б і в – програмовані переміщення робочого органу по осі 0X; г і д – програмовані переміщення робочого органу по осі 0Y

На рис. 6.22 прийняти наступні позначення: 1 і 2 – пневмоприводи обертової дії з програмованим циклом зміни кутів і напряму повороту ведучих ланок або два сервопривода; 3 і 4 – ведучі ланки (коромисла або кривошипи); 5 і 6 – шатуни; 7 – пневмоциліндр; 8 – робочий орган

(пневмоприсмоктувач) або робочий інструмент; 9 – державка робочого інструменту та пневмоциліндру; 10 – станина.

Державка 9 з'єднана з кінематичної парою, яка утворена головками шатунів 5 і 6 двокоромислового або двокривошипного механізму 3-5-6-4, а переміщення робочого органу (робочого інструменту) 8 програмується по трьом осям 0X, 0Y, 0Z.

Контрольні питання до розділу 6.

1. Які конструктивні особливості поршнів з різними типами демпфірування ?
2. Яке призначення і які конструктивні особливості поворотних пневмоциліндрів в схемах мехатроніки ?
3. Яке призначення і які конструктивні особливості без штокових циліндрів в схемах мехатроніки?
4. Яке призначення і які конструктивні особливості тандем циліндрів **і** опозитних циліндрів в схемах мехатроніки ?
5. Яке призначення і які конструктивні особливості мультіпозиційних циліндрів і комбінованих циліндрів в схемах мехатроніки ?
6. Яке призначення і які конструктивні особливості пневморозподільників в схемах мехатроніки ?
7. Яке призначення і які конструктивні особливості логічних пневмоклапанів в схемах мехатроніки ?
8. Яке призначення і які конструктивні особливості передавальних механізмів в мехатронних системах ?
9. Чим відрізняються важільні механізми з пневмоприводом на рис.6.17 ?
10. Для яких кутів повороту коромисел 3 і 4 зображені положення двокоромислового 3D-механізму з програмованими пневмоприводами на рис. 6.23,б … рис.6.23,д ?

7. ПРОГРАМОВАНІ ДАТЧИКИ, ТАЙМЕРИ, ЛІЧИЛЬНИКИ МЕХАТОННИХ СИСТЕМ ТЕХНОЛОГІЧНИХ МАШИН І ВЕРСТАТІВ

При проектуванні автоматизованих технологічних машин і верстатів використовують датчики, принцип роботи яких заснований на різних фізичних ефектах. В схемах мехатроніки технологічних машин і верстатів переважно застосовують датчики (сенсори) контроля механічних величин і насамперед кута повороту та похідних по часу від кута повороту та датчики лінійних переміщень - кінцеві вимикачі і датчики наближення (магнітні, індуктивні, ємкостні, оптичні, пневматичні, ультразвукові та інші) та датчики зусилля. В схемах мехатроніки технологічних апаратів переважно застосовують датчики (сенсори) контроля таких неелектричних величин, як температура, витрата рідини, рівень та концентрація рідини та інші. Тому інженер-конструктор або розробник при прийнятті рішення вибору типу датчиків керуються досвідом і знаннями, які пов'язані з наступними діями:

- вибором типу датчиків в залежності від функціонального призначення об'єкту проектування та умов його роботи у зовнішньому середовищі;
- вибором датчиків за технічними характеристиками і типу сигналів та їх обмеження на вході і на виході.

Сенсор – поширена назва «***датчик***» (*англ. sensor*) - це чутливий елемент для ***перетворення неелектричної величини*** з вихідними параметрами механічної природи (переміщення, швидкості, прискорення, сили, моменту в технологічних машинах і верстатах) або неелектричної величини іншої природи (температури, рівня, вологості, в'язкості та інших), які змінюються, контролюються або регулюються в технологічних апаратах ***в електричну величину*** для керування та сигналізації в схемах мехатроніки.

Наприклад, в технологічних CNC-машинах і CNC-верстатах машинобудування такими неелектричними величинами є кут повороту головного валу, лінійне переміщення повзуна або іншої ланки механізму, швидкість, прискорення, сили, обертаючий момент, наявність факту обриву ниток, кількість підрахованих штучних виробів або циклів та інші механічні і технологічні параметри, які змінюються або з'являються при роботі машини і які потрібні для реалізації запрограмованого циклу роботи вбудованого в машину контролера.

Цифрові піни контролера прописуються в програмі як входи (INPUT) та як виходи (OUTPUT), в тому числі і для ШІМ-синалів. В ПЛК цифрові входи і виходи прописуються в алокейшен листі програми. Аналогові сигнали на виході датчиків (температури, вологісті, тиску, концентрації розчинів, в'язкості рідини та інших неелектричних параметрів) подаються на аналоговие входи контролера. Перетворення аналогового сигналу датчика у дискретний (цифровий) сигнал виконує вбудована в контролер мікросхема аналого-цифрового перетворювача (АЦП).

7.1. **Аналого-цифрові перетворювачі в схемах мехатроніки**

Контролери, наприклад, поширені контролери лінійки Arduino можуть використовувати цифрові вхідні та вихідні сигнали і аналогові входні.

Аналоговий сигнал - це сигнал, який може приймати будь-яку кількість значень, у відмінності від цифрового сигналу, який має тільки два логічних значення: високий (1) і низький (0). Для вимірювання значень аналогових сигналів в контролері Arduino є вбудований **А**налого-**Ц**ифровий **П**еретворювач (**АЦП**, *англ.* **ADC – A**nalog-to-**D**igital **C**onverter). АЦП перетворює аналогову напругу з виходу датчика в цифрове значення. АЦП мікроконтролера ATMega328 має дозвіл $\mathbf{2^{10} = 1024}$ біта. Ці біти замінюються масивом зі 1024 значеннями, так званими **бінамі**, які представляють кожне з можливих значень АЦП. Такий процес апроксимації (заміни) має назву «**бінінг**». Так як доступна пам'ять обмежена, можуть бути створені біни тільки розміром в байт (8 біт). Тому число відкликів, обмежується числом $\mathbf{2^8 = 255}$. **Бін** (*англ.* ***bin*** - елемент дискретизації) - слово, що позначає деяку елементарну одиницю, неподільний елемент або частку мінімально можливого розміру. Іншими словами, АЦП здійснює перетворення аналогової безперервної величини в дискретну, тобто аналоговий сигнал датчика перетворюється в цифровий код, придатний для використання в контролері.

АЦП є важливим компонентом мехатронних і робототехнічних систем з датчиками, вбудованих у технологічні апарати, машини и верстати. Акселерометр, гіроскоп (гіротахометр), барометр, магнітометр, і навіть відеокамера - всі ці прилади з'єднуються з мікроконтролерів контролера за допомогою АЦП.

АЦП конструктивне може перебувати в одному корпусі з мікроконтролером (в одної інтегральної схемі), як у контролера Arduino Uno. Також АЦП можуть бути конструктивне виконані у вигляді окремої мікросхеми, наприклад MCP3008.

Пристрій зі зворотною функцією АЦП має назву **Ц**ифро-**А**налоговий **П**еретворювач (**ЦАП**, *англ.* **DAC – D**igital-to-**A**nalog **C**onverter), який переводить цифровий сигнал в аналоговий. Наприклад, під час програвання мелодії на мобільному телефоні відбувається перетворення цифрового коду з MP3 файлу в звук, який чуємо у себе в навушниках.

Контролер Arduino UNO не містить вбудованого **ЦАП** але Arduino використовує цифрові сигнали з широтно-імпульсною модуляцією (**ШІМ**) для реалізації функцій при роботі з аналоговим виходом. Для виведення (запису) **ШІМ-сигналу** використовувана функція (в дужках аргументи функції):

analogWrite (pin, value),

де **pin** – це номер виводу, що використовується для ШІМ виходу;

value – це значення числа, яке пропорційне коефіцієнту заповнення сигналу.

Коли **value** = 0, на виході завжди логічний «нуль». Коли **value** = 255, на виході завжди логічна «одиниця». На більшість плат Arduino, ШІМ функції доступні на виводах (пінах) з позначками (**~**): **~3, ~5, ~6, ~9, ~10** і **~11**. Частота ШІМ сигналу на більшості виводів складає приблизно 490 Гц. На Arduino Uno та подібних платах виводи 5 і 6 працюють на частоті приблизно 980 Гц.

Для порівняння аналогового вхідного значення **АЦП**, яке знаходиться в діапазоні від **0** до **1023** (рис. 7.1,д), з вихідним **ШІМ** сигналом, який знаходиться в діапазоні від **0** до **255** можливе використання наступної функції **map** (**аргументи**):

Функція	Аргументи				
map	аналогове значення	рівні *АЦП*		рівні *ШІМ*	
	(value,	**fromLow,**	**fromHigh,**	**toLow,**	**toHigh)**
		0,	1023	0,	255

Контролер Arduino Uno містить на своїй платі 14 цифрових входів (піни 0…13) загального призначення, до яких під'єднають світлодіоди і кнопки, 6 аналогових входів (A0…A5) та піни живлення.

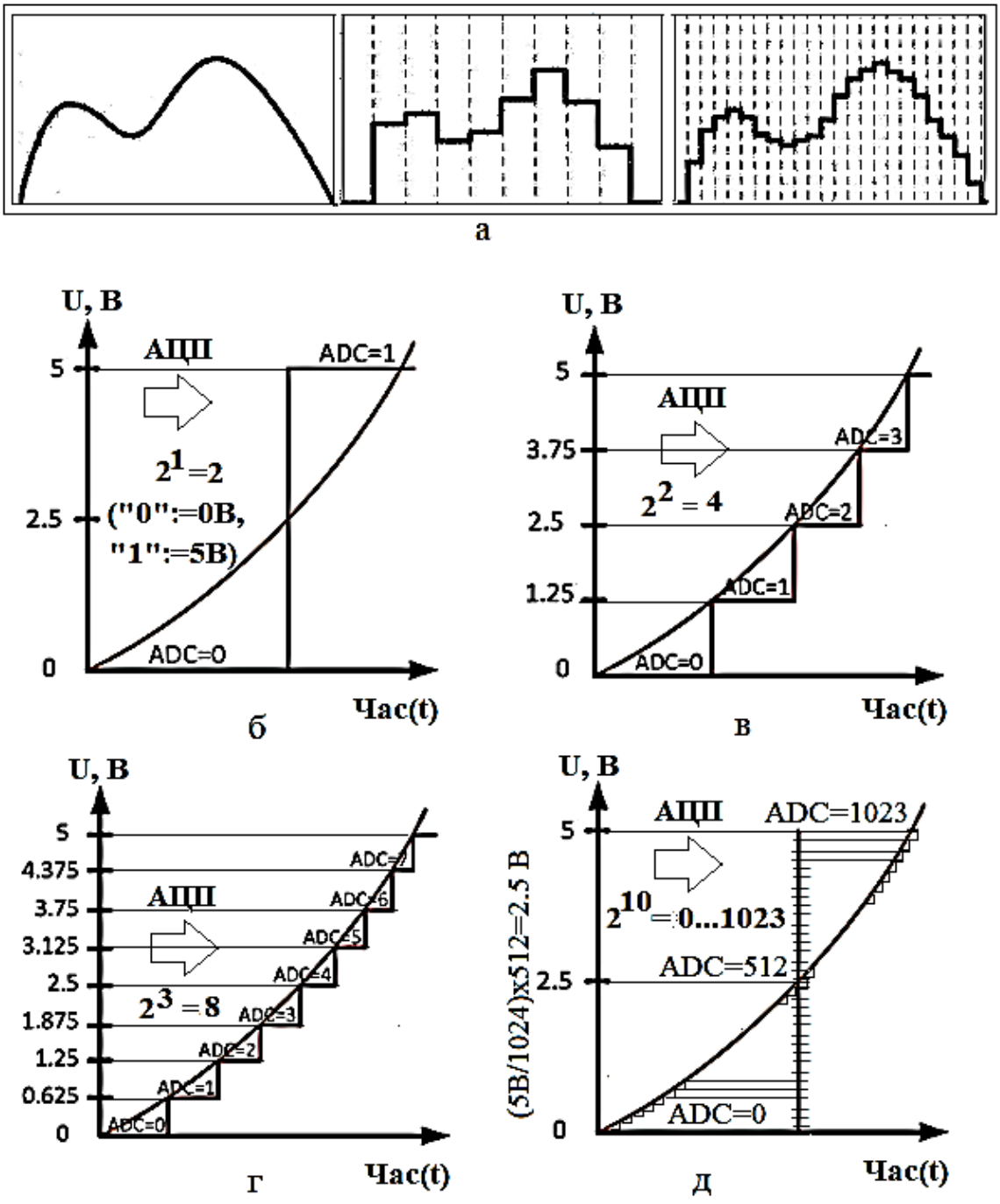


Рис. 7.1. Принципові схеми перетворення аналогового сигналу датчика у дискретний (цифровий сигнал) UNO: а – аналоговий сигнал датчика і цифрові сигнали після аналого-цифрового перетворення ; б – при

використанні 1-бітового АЦП; в – при використанні 2-бітового АЦП; г –при використанні 3-бітового АЦП ; д – при використанні 10ті-бітового АЦП контролера Arduino

В інших версіях Arduino аналогових входів може бути більше, наприклад, у **Arduino Mega** аналогових входів 16.

При роботі з кнопками та світловодами в програмі використовується функція **digitalRead()**, яка вміє зчитувати цифровий сигнал з певного цифрового входу (піну) контролера. При роботі з аналоговими датчиками в програмі використовується функція **analogRead()**, яка служить для зчитування аналогового сигналу з певного аналогового входу (піну) контролера.

Після виклику функції **analogRead(**номер піну**)** мікроконтролер контролера виміряє рівень аналогового сигналу на заданому аналоговому номері піну для АЦП і зберігає результат роботи АЦП в змінну «результат». При цьому результатом функції **analogRead** буде число від 0 до 1023 (рис. 7.1,д).

Розрядність АЦП. Треба зауважити, що число 1023 тут з'явилося неспроста. Справа в тому, що у кожного пристрою АЦП є такий важливий параметр як розрядність. Чим більше значення цього параметра, тим точніше працює прилад. Припустимо, що у нас є АЦП з розрядністю 1 (рис. 7.1,б). Подаючи на вхід будь напруги від 0 до 2,5 Вольт, на виході ми отримаємо логічний «0». Будь-яке ж напруга від 2,5 до 5 вольт дасть нам логічну «1». Тобто 1-бітний АЦП зможе розпізнати тільки два рівня напруги.

АЦП з розрядністю 2 (рис. 7.1,б) розпізнає вже чотири рівні ($2^2 = 4$) напруги: від 0 до 1,25 - це 0; від 1,25 до 2,5 - це 1; від 2,5 до 3,75 - це 2; нарешті, від 3,75 до 5 - це 3.

АЦП з розрядністю 3 (рис. 7.1,в) розпізнає вже вісім рівнів ($2^3 = 8$) напруги.

Контролер **Arduino Uno** містить 10-бітний (10-розрядний) АЦП і це означає, що будь-яка напруга на аналоговому вході в діапазоні від 0 до 5 вольт буде перетворено в число з **точністю = 5В/1024** $\approx$ 0.005 В = 5 мВ (5 мылывольт). На графіку складно зобразити 1024 сходинок з кроком 5 мВ (рис. 7.1,д) для апроксимації аналогового сигналу цифровим сигналом.

Таким чином, маючи таку точність, 10-бітний АЦП може «відчути» від датчика зміну напруги на вході всього у 5 мілівольт.

Опорна напруга. Є нюанс, який може стати причиною помилки вимірювання за допомогою АПЦ. Діапазон від 0 до 5 вольт, в якому працює пристрій у загальному випадку виглядає інакше, а саме від **0** до **опорної напруги**. Ця зміна уточнює формули для розрахунку точності АЦП:

точність АЦП = опорна напруга / 1024

Опорна напруга визначає межу діапазону, з яким буде працювати АЦП.

Наприклад, якщо живлення контролера відбувається від чотирьох Ni-Cd акумуляторів на 1.2 Вольта. Тоді точність вимірювання буде пов'язана з кроком 4.8В /1024 = 4,7 мВ і це слід врахувати в програмі.

Якщо Arduino Uno в якості опорної напрузі буде обрано, наприклад, напруга 3.3 Вольта, тоді використовують спеціальний пін (контактний штирь) **AREF**. На цей пін потрібно подати напругу 3.3 Вольта, а дозвіл на використання зовнішнього джерела опорної напруги забезпечується за допомогою функції **analogReference (EXTERNAL)**, яку слід викликати всередині функції **void setup()** програми.

Також слід враховувати, що результат вимірювання значення напруги не може перевищувати межі діапазону. Якщо ми вибираємо в якості опорного напруги 3.3 Вольта, а надходить сигнал буде з більшою електричною напругою, то ми отримаємо неправильне значення напруги, оскільки АЦП «не знає» про наявність більш високої напруги.

Наведений приклад програми з використанням АЦП після її компіляції буде кожну секунду вимірювати аналогове значення на вході A0, і передавати його в послідовний порт (COM-порт) командою **`Serial.begin(9600)`**.

```
int val = 0;
void setup() {
    Serial.begin(9600);
    pinMode(A0, INPUT);
}
void loop() {
```

```
    val = analogRead(A0);
    Serial.println(val);
    delay(1000);
}
```

7.2. **Дискретні датчики кута повороту ланок програмно керованих механізмів та машин в схемах мехатроніки**

Енкодер. Для автоматичного контролю кута повороту і положення ротора двигуна або головного валу технологічної машини використовується датчик кута повороту *е н к о д е р.* ***Енкодер*** це цифрової сенсор для визначення швидкості обертання, наприклад, ротору сервоприводу технологічних CNC-машин та CNC-верстатів машинобудування. На виході енкодера формується двофазна послідовність імпульсів, співвідношення фаз яких характеризує напрямок обертання та частоту обертання, яка пропорційна частоті цих імпульсів. При цьому кількість імпульсів пропорційна кількості скоєних оборотів або величині лінійного переміщення в *мм* (mm) чи в *дюймах* (inch) робочого органу (робочого інструменту) машини/верстату на один оборот ротора з урахуванням передаточного відношення.

Для обчислення кутової швидкості об’єкта процесор енкодера виконує диференціювання кількості імпульсів у часі, таким чином, показуючи відразу величину кутової швидкості, тобто число оборотів в хвилину. Вихідний сигнал має два канали, в яких ідентичні послідовності імпульсів зрушені на 90° відносно один одного, що дозволяє визначати напрямок обертання.

Енкодери мають широке застосування для зворотного зв’язку в технічних засобах робототехніки, металообробці, ліфтової техніки, автоматах для фасування, пакування і розливу, в випробувальних стендах, а також в інших CNC-машинах та CNC-верстатах з програмованими приводами чи приводом, що вимагають точної реєстрації показників руху робочого органу чи робочого інструменту. Енкодери замінили широко поширені раніше сельсини.

Основна характеристика енкодера - кількість імпульсів (рисок) на оборот (PPR)-Pulses Per Revolution. Контролер може працювати з енкодером до 4000 імпульсів /оборот. Ще одна величина - кількість відкликів на оборот, яка в 4 рази більше ніж PPR, контролер у своїй роботі

оперує саме цією величиною. Енкодери зазвичай мають два виходи: вихід А і вихід В для того щоб визначити - обертається вал за годинниковою стрілкою або проти годинникової стрілки, засноване на зсуві фази 90° . Енкодер з одним виходом (А) більш відомий як тахометр. Для підвищення перешкодозахищеності використовуються енкодери з так званим диференціальним виходом, який на кожну з фаз містить по два виходи, що позначаються як А +, А-, В +, В- .

Інкрементний енкодер формує імпульси, кількість яких відповідає повороту валу на певний кут. Цей тип енкодерів, на відміну від абсолютних енкодерів, не формує код положення валу, коли вал знаходиться в спокої.

Абсолютний енкодер відноситься до типу енкодерів, який виконує унікальний код для кожної позиції валу. На відміну від інкрементного енкодера, лічильник імпульсів не потрібен, тому що кут повороту завжди відомий.

Абсолютний енкодер формує сигнал як під час обертання, так і в режимі спокою. Кодовий диск абсолютного енкодера (рис. 7.2) відрізняється від диска покрокового енкодера, оскільки має кілька концентричних прозорих доріжок з непрозорими ділянками по колу. Кожна доріжка формує унікальний двійковий код для конкретної позиції (кута повороту) валу, на якому закріплений кодовий диск (рис. 7.2).

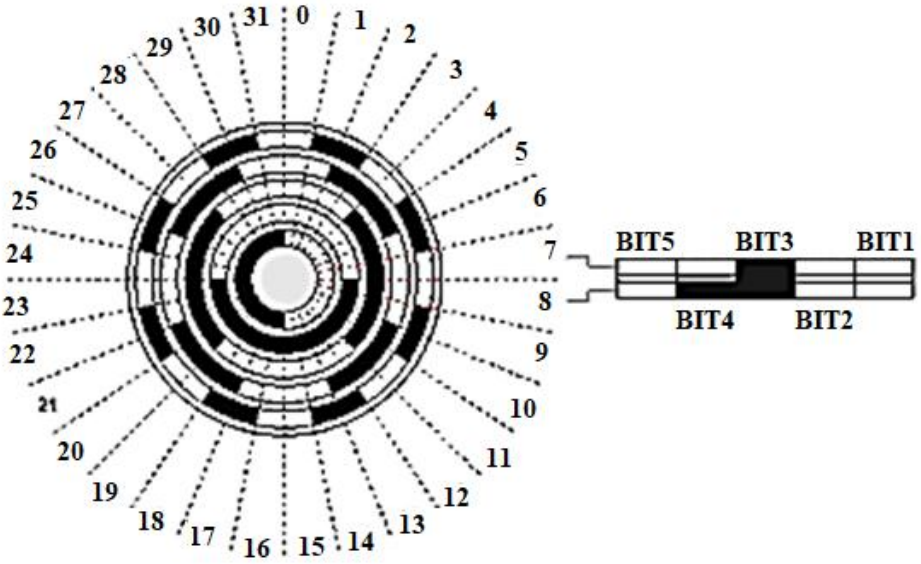


Рис. 7.2. Кодовий диск абсолютного енкодера

Абсолютний енкодер не втрачає свого значення при втраті живлення і не вимагає повернення в початкову позицію. Сигнал абсолютного енкодера не схильний перешкод і для нього не потрібна точна установка валу. Крім того, навіть ЯКЩО кодований сигнал не може бути прочитаний енкодером, ЯКЩО, наприклад, вал обертається дуже швидко, правильний кут обертання буде зареєстрований, коли швидкість обертання зменшиться. Абсолютний енкодер стійкий до вібрацій.

Датчикі кута повороту валу, коромисла, кривошипу або ексцентрика механізмів технологічних машин застосовуються у мехатронних системах для регістрації або контролю кутового і лінійного положення, напряму руху і частоти обертання ведучих або ведених ланок цільових та виконавчих механізмів. Такі Датчикі абсолютного значення перетворюють кут повороту валу в електричний сигнал, який є певним цифровим кодом, наприклад двійковим, двійкове-десятинним або *кодом Грея* (табл. 7.1). Число розрядів коду визначає точність регістрації кутового положення валу. 8-розрядний код (2^8=256) дозволяє регіструвати кут повороту з точністю 0,7 градуса $\frac{256}{360} = 0{,}7^0$.

Код Грея це одна із систем кодування інформації, в якій *два послідовні коди* відрізняються значенням лише *одного біта*, тобто зміна кодованого числа на одиницю відповідає зміні кодової комбінації тільки в одному розряді, згідно з наведеною таблицею.

Код Грея переважає двійкововий код тим, що має властивість безперервності бінарної комбінації, а саме зміна кодованого числа на одиницю відповідає зміні кодової комбінації тільки в одному розряді. Код Грея будується за наступним правилом: старший розряд залишається без зміни; кожний наступний розряд інвертується, ЯКЩО попередній розряд вихідного двійкового коду дорівнює одиниці. Цей алгоритм побудови може бути представлений як результат складання за модулем двох вихідних комбінацій двійкового коду з такою ж комбінацією кодів, але зрушеної на один розряд вправо. При цьому крайній правий розряд зрушеною комбінації відкидається.

Таким чином, код Грея є так званим одно кроковим кодом, тому при переході від одного числа до іншого завжди змінюється лише якийсь один біт. Похибка при зчитуванні інформації з механічного кодового диска при переході від одного числа до іншого призведе лише до того, що перехід від одного положення до іншому буде лише дещо зміщений за часом,

однак видача абсолютно невірного значення кутового положення при переході від одного положення до іншого повністю виключається .

Напрямку обертання валу визначається здатністю Грей-коду дзеркального відображення інформації, якщо інвертувати старший біт коду.

Оскільки інформація, виражена в Грей-коді, має чисто кодований характер що не несе реальної числової інформації, код повинен перед подальшою обробкою спершу перетворений в стандартний бінарний код. Здійснюється це за допомогою перетворювача коду (декодера), який реалізується за допомогою ланцюга з логічних елементів «виключає або» (XOR) як програмним, так і апаратним способом.

Таким чином, кодом Грея називають код, який реалізується за допомогою двох символів 0 та 1 при зростанні цілого числа за наступним алгоритмом:

1 – наймолодший 1-ий розряд у послідовності символів змінюють у такій послідовності: «0», «1», після чого у наступний 2-й розряд записують «1», а наймолодший розряд змінюють уже у протилежному порядку;

2 – два наймолодші розряди змінюють у такому порядку: **00, 01, 11, 10**, після чого у 3-ій розряд записують 1 і два наймолодші розряди змінюють уже у протилежному (дзеркальному) порядку:**10, 11, 01, 00**;

3 – три наймолодші розряди змінюють у такому порядку: **000 , 001, 011, 010, 110, 111, 101, 100**, після чого у 4-ий розряд записують 1 і три наймолодші розряди змінюють уже у протилежному (дзеркальному) порядку: **100, 101, 111, 110, 010, 011, 001, 000 ... ;**

4 – наступні наймолодші k-розряди змінюють у порядку, визначеному попередніми кроками, після чого у наступний (k + 1) розряд записують 1, а молодші розряди змінюють уже у протилежному порядку і т. д.

У таблиці 7.1 подано двійникові коди і коди Грея для 4-розрядної послідовності чисел від 0 до 15 включно.

Таблиця 7.1 Двійникові коди і коди Грея

10-система	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2-ий код	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Код Грея	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000

.

Таким чином, як випливає з наведеної таблиці код Грея двох послідовних натуральних чисел відрізняються лише в одному розряді. Код Грея використовують для збільшення надійності роботи системи оптичних фотодіодів при встановленні кута повороту дисків-носіїв інформації.

З таблиці 7.1 випливає, що при переході від одного числа до іншого (сусіднього) лише один біт інформації змінює свій стан, У двійковому коді можуть поміняти свій стан кілька біт одночасно. Код Грея не має помилки читання і тому застосовується в багатьох абсолютних енкодерах.

Звичайний одно крокoвий Грей-код підходить для розрішеній, які можуть бути представлені у вигляді числа зведеного в ступінь 2. У випадках, де треба реалізувати інші розрішення зі звичайного Грей-коду, вирізається і використовується середня його ділянка. Таким чином, зберігається «однокроковість» коду. Однак числовий діапазон починається не з нуля, а зміщується на певне значення. При обробці інформації від генеруючого сигналу віднімається половина різниці між початковим і редукованим дозволом. Наприклад, для виразу кута 360° в 9-ти бітному Грей-коді $2^9 = 512$ кроків, Грей-код урізується з обох сторін на 76 кроків (512 - 2*76 = 360) що дорівнюється 360° – одному обороту головного валу технологічної машини.

Імпульсний датчик кута повороту валу (енкодер) складається з кодового диску, який закріплений на валу, а світлодіоди і фото транзистори закріплені на корпусі машини по різні сторони кодового диску напроти друг другу (рис. 7.3,б). Кодовий диск поділений певним

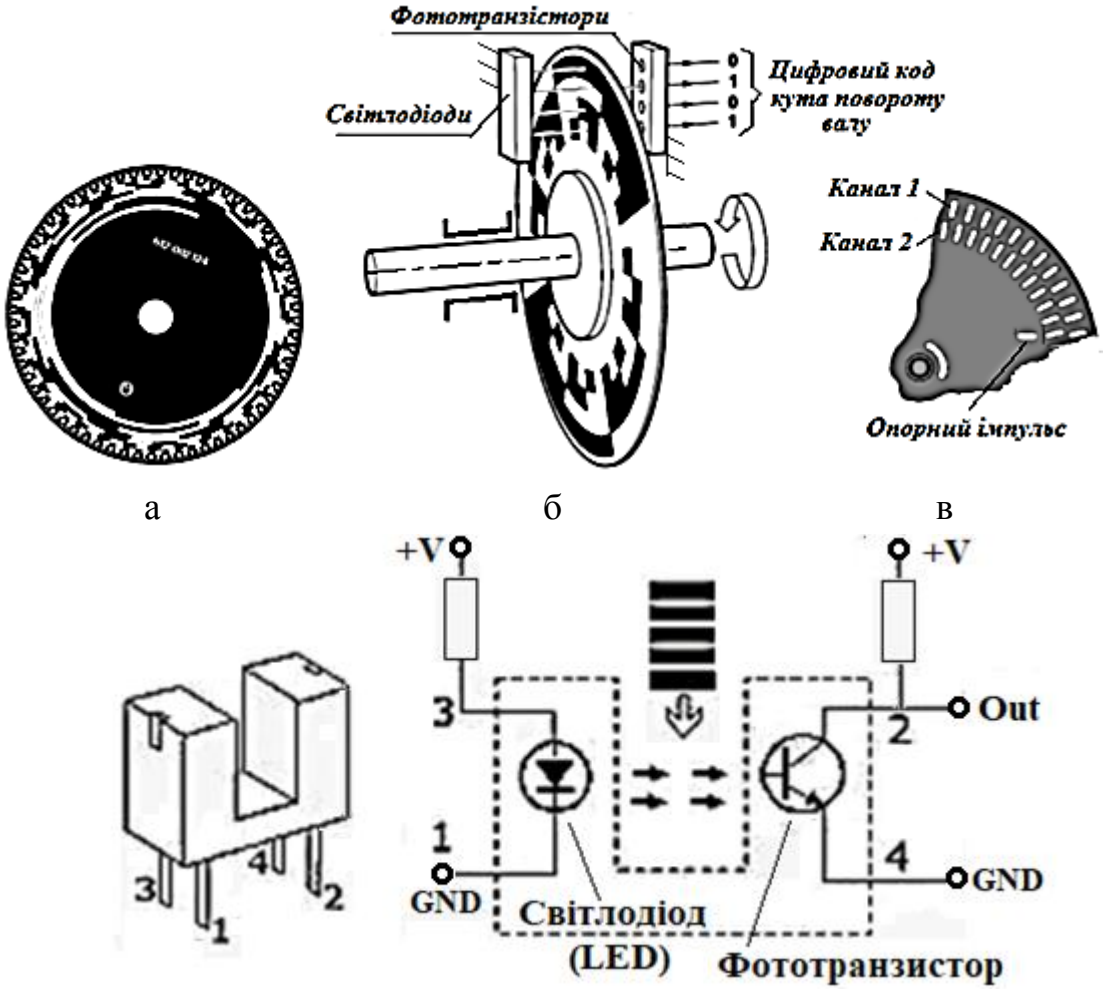


Рис. 7.3. Принципові схеми оптичного енкодера технологічних машин: а – кодовий диск з кодом Грея; б – датчик абсолютного значення кута з кодовим диском на валу; в – фрагмент кодового диску імпульсного датчика кута повороту валу; г – загальний вигляд і схема підключення імпульсного датчика до контролера

чином на прозорі і непрозорі зони для світла світлодіодів. Кожному куті повороту валу відповідає унікальне сполучення прозорих і непрозорих ділянок, які за допомогою фототранзисторів перетворюються в логічні «1» і «0». Фототранзистори видають інформацію о положенні диску у вигляді двійкового коду через кожні 0,7 градуса кута повороту валу. Таким чином імпульсний датчик кута повороту валу перетворює кут повороту валу в послідовність електричних імпульсів (а ні в цифровий код Грея), які підраховуються і визначається сумарний кут повороту валу. Точність, з якої датчик визначає зміну кута повороту залежить від числа імпульсів на

один оборот валу. Наприклад, при 1000 імпульсів на 1 оборот зміна кута регіструється з точністю 360/1000 = 0,36 градусів.

Імпульсний датчик положення, як і в датчик абсолютного значення кута має кодовий диск. На кодовому диску-діафрагмі фотоспособом нанесені прорізи в два ряди (*канал 1* і *канал 2* на рис. 7.3,в), при цьому прорізи одного каналу зсунені відносно прорізів другого каналу на пів імпульсу. Якщо визначити від якого каналу імпульси приходять першими, можна визначити напрям обертання валу за стрілкою годинника або проти стрілки годинника. У третьому каналі є проріз що служить для опорного або так званого «нульового» імпульсу, який зчитується один раз з один оберт валу. Якщо на кодовому диску нанесені прорізи для двох опорних імпульсів зсунутих по фазі на кут π радіан, то такий імпульсний датчик може бути застосований в швейної машині зі стоп-мотором для автоматичного зупинення головного валу по першому опорному імпульсу при знаходженні голки у крайньому верхньому положення над матеріалом, а по другому опорному імпульсі виконується зупинення головного валу при знаходженні голки в матеріалі при її крайньому нижньому положенні. При крайньому верхньому положенні голка надає команду на включення механізму обрізки ниток в кінці шва або в кінці циклу автоматичного пришивання фурнітури або виготовлення петлі на одязі. При крайньому нижньому положенні голки можна подавати команду від педалі (натискання педалі п’яткою ступні ноги) для автоматичного переміщення вверх притискної лапки для наступного перехвату (повороту деталей округ осі голки) при контурному виконанні ниткових швів.

Таким чином енкодери створюють прямокутні імпульси, що характеризують рух за короткий проміжок часу. При автоматичному підрахунку цих імпульсів визначається кут повороту або похідна від кут повороту по часу (кутова швидкість) ротора, а порядок чергування імпульсів по двох каналах визначає напрямок обертання. Для отримання точних позицій і швидкості ротора приводів можуть також застосовуватися датчики у вигляді резольвера.

Аналоговим датчиком кута повороту валу є **резольвер**, який генерують послідовність синусоїдальних і косинусоїдальних імпульсів аналогового напруги, що характеризують абсолютне положення ротора за один оборот. Ці аналогові сигнали зазвичай перетворюються в цифрові

інтерфейсною платою резольвера. Практично, всі сучасні приводи машин і верстатів змінного і постійного струму допускають використання енкодерів в лінії зворотного зв'язку, і лише для невеликої частини з них можуть бути застосовані резольвери.

Датчик Холла. Безконтактний магнітоелектричний датчик кута повороту головного валу технологічних машин та/або частоти його обертання. Датчик має щілину, по одну сторону якої розташована напів-провідникова прямокутної форми пластина, а з протилежної сторони щілини змонтований постійний магніт. В щілину сенсора входить сталевий кодовий диск з пелюстками, який при роботі машини обертається і його пелюстки перетинають магнітне поле постійного магніту.

Сервопривод з внутрішнім енкодером зворотного зв'язку 360^0 має постійний магніт усередині, який прикріплений до валу ротора (рис. 7.4). В корпусі статора знаходиться датчик Холла. Магніт змінює своє положення при обертанні валу двигуна і датчик Холла реагує на зміни положення вектору магнітного поля магніту. Мікропроцесор контролера постійно реагує на кут повороту магніту і його результат у вигляді ефекту Холла. Датчик Холла серводвигуна відправляє отриману цифрову інформацію про кут повороту на контакт (пін), наприклад, контролера Arduino або іншого, до якого цей датчик підключений.

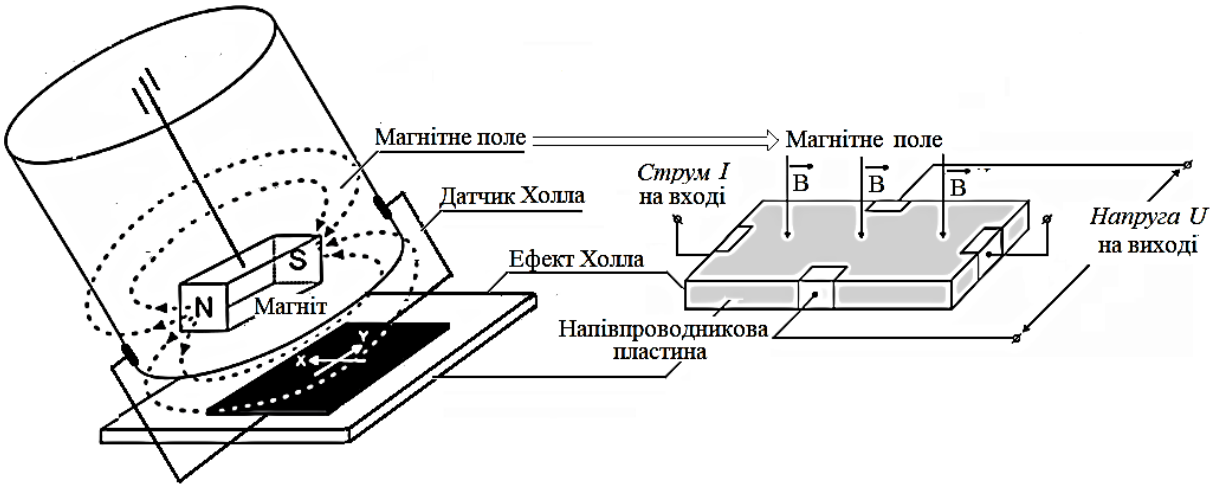


Рис. 7.4. Схема фізичного принципу роботи сенсора Холла: а – принципова схема серводвигуна з датчиком Холла; б – схема ефекту Холла

Принцип роботи сенсора заснований на фізичному ефекті Холла, а саме в існуванні наступного фізичного ефекту: **«ЯКЩО до протилежних сторін напівпровідникової прямокутної за формою пластинки підвести джерело постійного струму I та одночасно піднести постійний магніт, що створює зовнішнє магнітне полі з магнітною індукцією B (рис. 7.4) і як результат на другій парі сторін напівпровідникової пластинки виникає різниця електричного потенціалу (е. р. с. Холла) або електрична напруга U»** Цей електричний сигнал знімається із сенсора, який з'являється з частотою перетинання, наприклад, пелюстками сталевого кодового диску (на рис. не зображений).

Приклади застосування сервоприводів з датчиками Холла наведені на рис. 7.5…7.7 у мехатронному (роботизованому) колесному рухомому об'єкті. Для програмування кутів повороту сервоприводів з внутрішніми енкодерами двох колес рухомого об'єкту в мехатроніці і робототехніці введений термін «тік», який означає 1/64 оборот колеса (рис. 7.5). Якщо колесо виконало один повний оборот на 64 тика то і контролер отримує від енкодеру 64 імпульсів. При довжині кола 208 мм колеса за 1 тик переміщується на відстань 208 мм/64 = 3.25 мм = 0.00325 м. Тому програмована відстань S руху в м рухомого об'єкту з індивідуальними сервоприводами колес визначається за формулою:

$$S = 0.00325 \cdot n_{\text{т}}, \qquad (7.1)$$

де $n_{\text{т}}$ – кількість тіків.

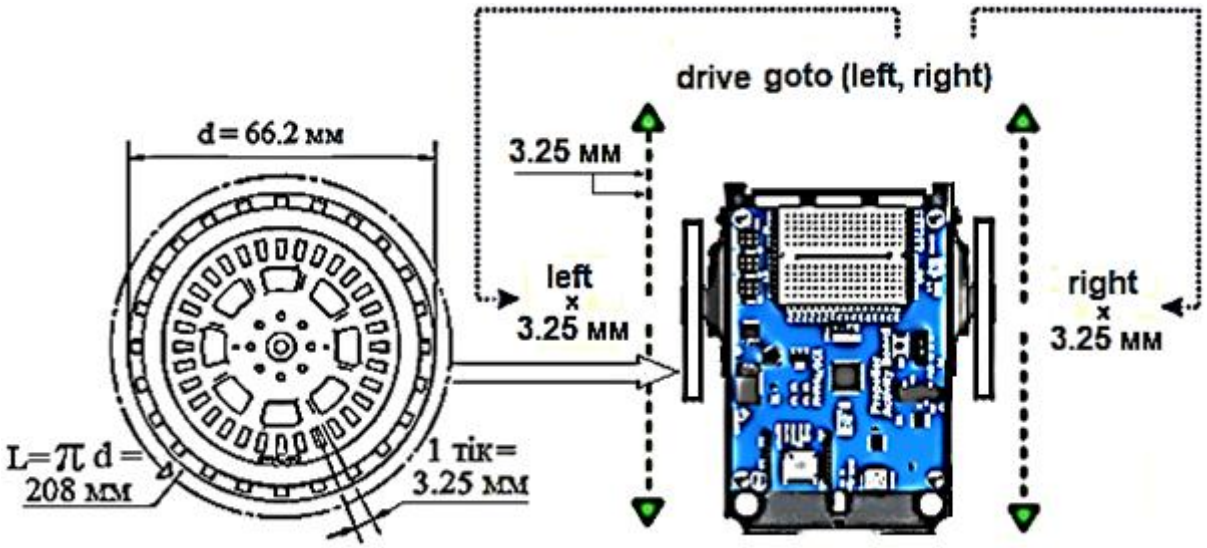
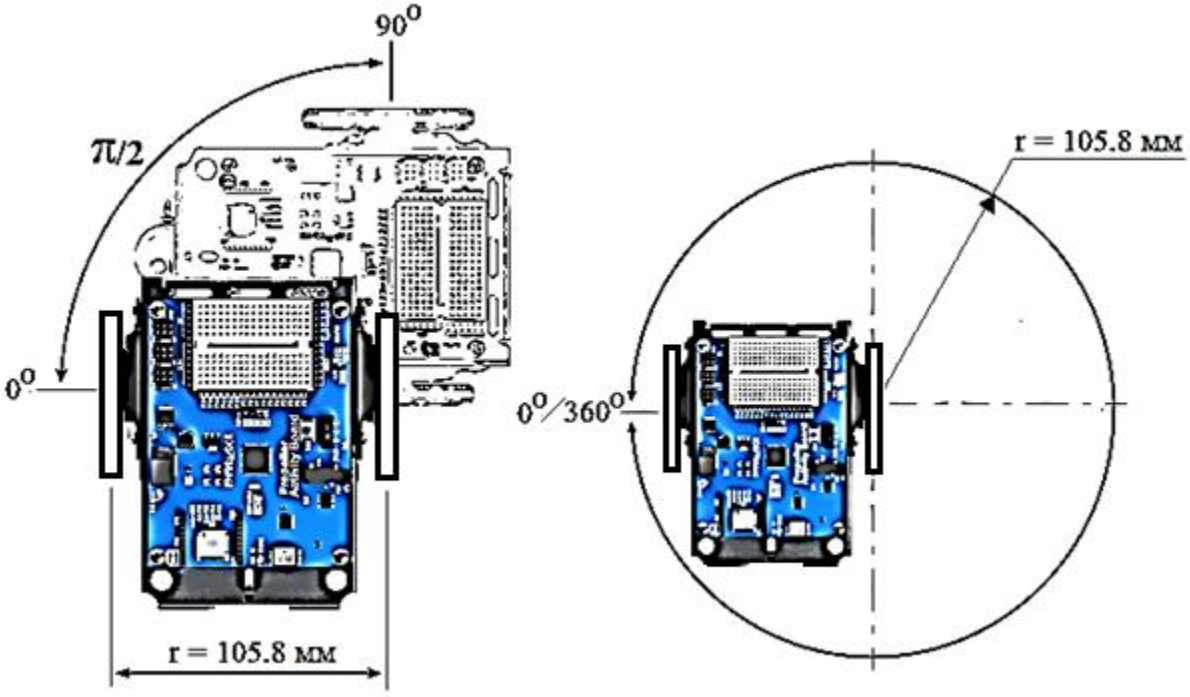


Рис. 7.5. Розрахункова схема та фрагмент коду для програмування кутів повороту двох сервоприводів з внутрішніми енкодерами колес рухомого об’єкту



а б

Рис. 7.6. Розрахункові схема для програмування кутів повороту рухомого об’єкту з двома сервоприводами колес : а – поворот вправо за стрілкою годинника на кут 90^0 ; б – поворот на кут 180^0

Для повороту на площині рухомого об’єкту (рис. 7.6) за стрілкою годинника на кут 90^0 і наступному руху вправо сервопривод лівого колеса програмно повертає його на 64/2 тіка при радіусі повороту r = 105,8 мм, а сервопривод правого колеса залишається нерухомим. Розрахункова схема для руху вліво після повороту наведена на рис. 7.6,а .

Для програмного повороту на кут 180^0 для руху об’єкту в протилежному напряму розрахункова схема наведена на рис. 7.6,б.

Для схем на рис. 7.6,а і на рис. 7.6,б кількість програмних тіків розраховуємо по формулами:

$$n_{\text{т}} = \frac{\pi \cdot r}{2\text{тіка}} = \frac{3.14 \cdot 105.8}{2 \cdot 3.25} \approx 51\text{тік} \qquad (7.2)$$

$$n_{\text{т}} = 2\pi \frac{r}{\text{тік}} = \frac{6.28 \cdot 105.8}{3.25} \approx 205 \text{ тіків} \qquad (7.3)$$

Розрахункова схема і фрагмент коду для програмування швидкості рухомого об’єкту з індивідуальними сервоприводами двох колес наведені на рис. 7.7.

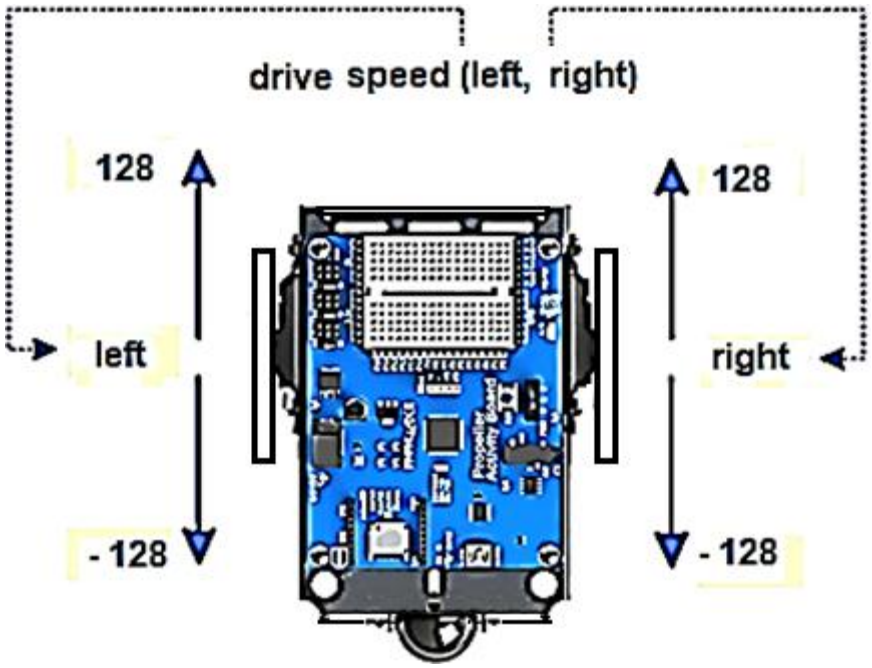


Рис. 7.7. Розрахункова схема та фрагмент коду для програмування швидкості рухомого об’єкту з двома індивідуальними сервоприводами колес з внутрішніми енкодерами

Розглянемо приклад програмування різної швидкості рухомого об’єкту на рис. 7.7 на двох ділянках шляху із зупинками та одним поворотом.

Цикл руху наступний:

на першої ділянці повинен відбуватися рух «вперед» зі швидкістю 64 тік/с і в кінці ділянці зупинитися;

на другої ділянці зробити поворот вправо на кут 45 градусів і зупинитися, наприклад зустрілася перешкода і ультразвуковий датчик дальності надає команду рухомому об’єкту обійти перешкоду. Для визначення такої перешкоди рухомий об’єкт оснащується ультразвуковим датчиком дальності, принцип дії і програмування якого розглянуто в підрозділі 7.6;

на третій ділянці після повороту відбувається знову рух «вперед» але зі швидкістю в два рази більшої ніж на першій ділянці (128 тік/с), в кінці ділянці треба зупинитися.

Тоді скетч буде мати наступний вигляд:

```
#include "simpletools.h" // виклик бібліотеки simpletools
#include "abdrive.h"     // виклик бібліотеки abdrive
 int main ()
 {
   Drive speed (64, 64); // рух прямо, 64 тік/с на протязі 2х секунд
   pause (2000);                    // пауза 2 с
   drive speed (0, 0);
   drive speed (26, 0);// поворот лівого колеса, швидкість 26 тік/с
   pause (1000);                    // пауза 1 с
   drive speed (0, 0);
   drive speed (128, 128); // рух прямо, швидкість 128 тік/ с
   pause (1000);                    // пауза 1 с
   drive speed (0, 0);
 }
```

Пояснення до наведеної програми.

Функція **drive speed (64, 64)** забезпечує рух прямо зі швидкістю 64 тік/с на протязі двох секунд.

Функція **drive speed (26, 0)** повертає ліве колесо зі швидкістю на 26 тіків/с на кут $45^O = \frac{\pi}{4}$ при цьому праве колесо нерухомо. При цьому 26 тіків отримуємо за формулою (7.2):

$$n_{\text{т}} = \frac{\pi \cdot r}{4\text{тіка}} = \frac{3.14 \cdot 105.8}{4 \cdot 3.25} \approx 26 \text{ тік} \qquad (7.4)$$

Функція **drive speed (128, 128)** забезпечує рух прямо зі швидкістю 64 тік/с на протязі одної секунди.

Функція **drive speed (0, 0)** зупиняє рухомий об’єкт.

Для розрахунку відстані і переміщення S_1 і S_3 рухомого об’єкту по отриманим даним для швидкості для двох ділянок шляху і відстані переміщення S_2 лівого колеса при повороті отримуємо:

$$S_1 = 64\ \frac{\text{тік}}{\text{с}} \times 2\ с \times 0.325\ \text{см} = 41.6\ см\ ;$$

$$S_2 = 26\ \frac{\text{тік}}{\text{с}} \times 1\ с \times 0.325\ \text{см} = 8.45\ см\ ;$$

$$S_3 = 128\ \frac{\text{тік}}{\text{с}} \times 1\ с \times 0.325\ \text{см} = 41.6\ см\ .$$

7.3. **Безконтактні кінцеві вимикачі**

До програмно керованих виконавчих механізмів зворотно-поступової дії, які застосовуються в технологічних машинах галузі і верстатах машинобудування відносяться: електромагнітні реле і соленоїди ; пневмоциліндри і гідроциліндри ; лінійні електродвигуни і серводвигуни; крокові двигуни.

Для автоматичного контролю і сигналізації крайніх положень переміщення вихідних ланок виконавчих механізмів зворотно-поступової і обертової дії використовуються контактні (механічні) в релейно-контактних схемах і безконтактні (електронні) кінцеві вимикачі.

Для реалізації автоматизованих режимів роботі циклових систем мехатроніки з контролером широко використовуються безконтактні кінцеві вимикачі різних типів, умовні позначення яких в електричних релейних схемах наведені на рис. 7.8. Електричні схеми підключення цих датчиків наведені на рис. 7.9.

Якщо на лабораторному стенді встановлені геркони з 2-дротовим кабелем і потрібно ***з а в ж д и*** *дріт червоного кольору* підключати до контакту **«+24В»** джерела струми, *дріт синього кольору* «**мінус**» підключати до котушки **К** реле, а другий контакт реле підключають на «**0В**» джерела струму (рис.7.9,а).

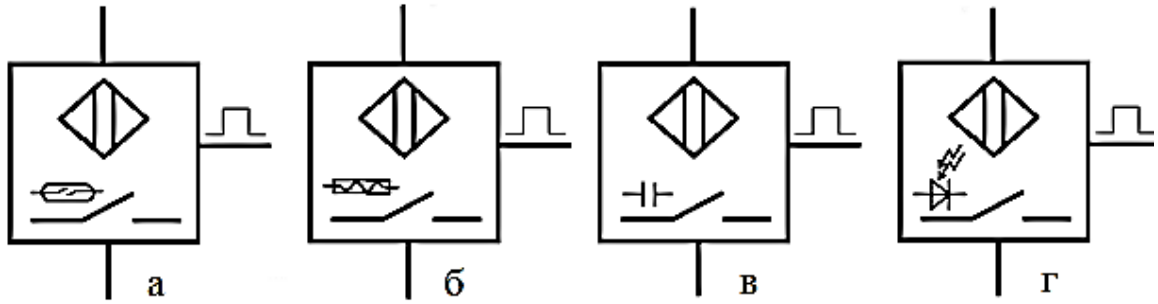


Рис. 7.8. Умовні позначення 3-дротових безконтактних кінцевих вимикачів : а –магнітний на базі геркона; б – індуктивний; в – ємкісний; г – оптичний

Оптичний і ємкісний датчики з 3-дротовим кабелем підключають також до контакту **«+24В»** джерела струми *дротом червоного кольору*, *дріт жовтого кольор*у датчика до котушки **К** реле, а *дріт синього кольору* і вихід котушки реле підключають на **«мінус»** (рис. 7.9,б).

Індуктивний датчик підключають наступним чином: *дріт червоного кольору* до **«+24В»** джерела струму, *дріт жовтого кольору* до котушки **К** реле і дріт синього кольору на **«мінус»** (рис. 7.9,в).

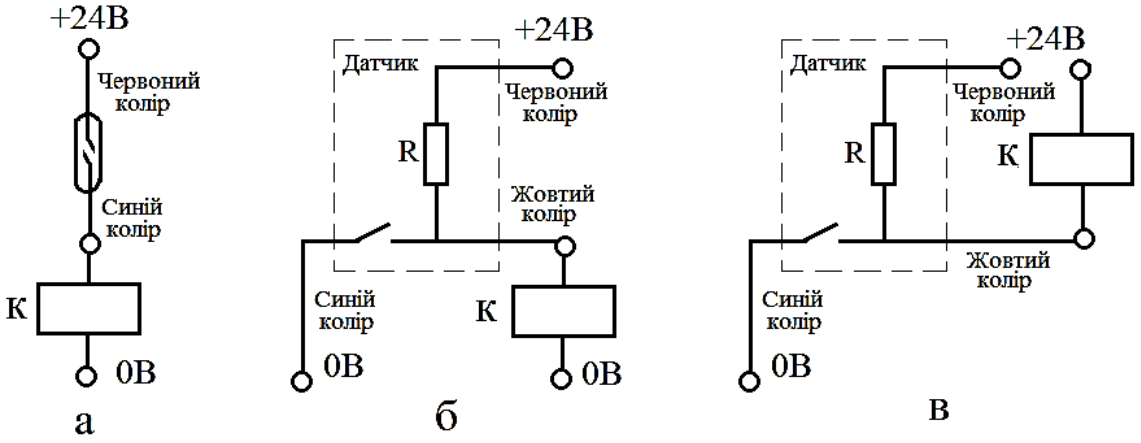
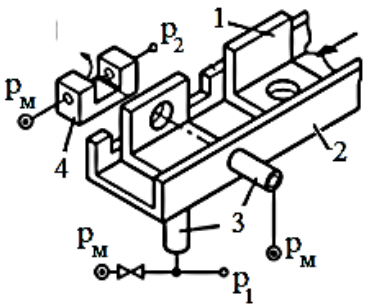
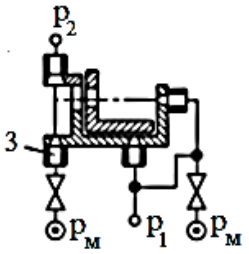


Рис. 7.9. Електричні схеми підключення безконтактних кінцевих вимикачів з двома і трьома кольоровими проводами : а – геркона; б – ємкісного і оптичного; в – індуктивного

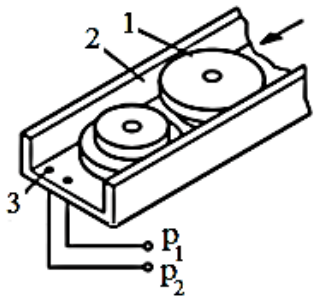
Приклади застосування безконтактних струменевих датчиків наведені гна рис.7.10. Для контроля правильної орієнтації деталей перед їх завантаження для наступної обробці або зборки використовуються струменеві датчика з контролем їх логічних сигналів (рис. 7.10). На рис. 7.10,а наведений процес контролю розташуванні отвору на деталі по кутовий форми і таблиця логічних сигналів, які пояснюють операцію орієнтації



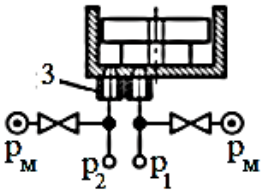
p_1	p_2	Результат контролю
0	0	Деталь відсутня
1	1	Деталь з браком
1	0	Невірна орієнтація деталі
0	1	Правильна орієнтація деталі



а



p_1	p_2	Результат контролю
0	0	Деталь відсутня
1	0	Деталь з браком
0	1	Невірна орієнтація деталі
1	1	Правильна орієнтація деталі

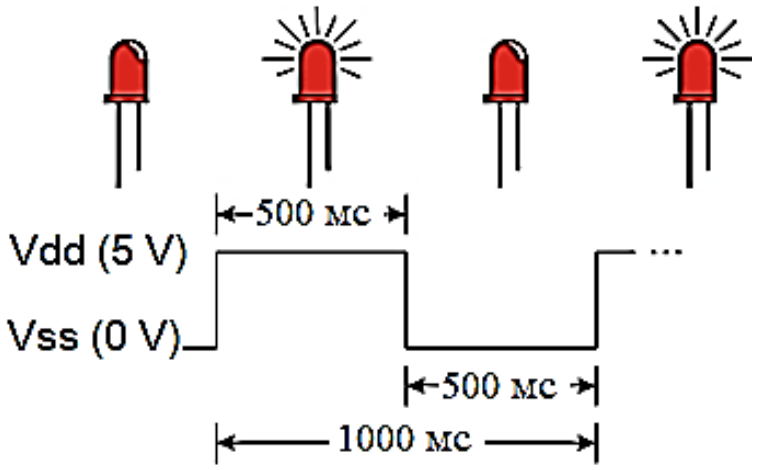


б

Рис. 7.10. Приклад застосування безконтактних струменевих датчиків положення деталей для контролю їх орієнтації (верх/низ) і таблиці програмованих логічних сигналів: а – деталей кутової форми з отвором; б – деталей циліндричної форми

7.4. **Програмування таймерів мовою STL мехатронних систем технологічних машин та верстатів**

При використанні контролера в схемах мехатроніки реле часу має назву таймер. В програмах керування таймери, лічильники і елементи пам’яті задаються інформаційно. В релейно-контактних схемах мехатроніки (без контролера) використовують реле часу у вигляді фізичного елементу. Приклад затримки включення реле часу по передньому фронту сигналу (із затримкою включення) і реле часу для затримки виключення по задньому фронту сигналу (із затримкою виключення) наведені, відповідно, на рис. 7.11 і на рис. 7.12.



а

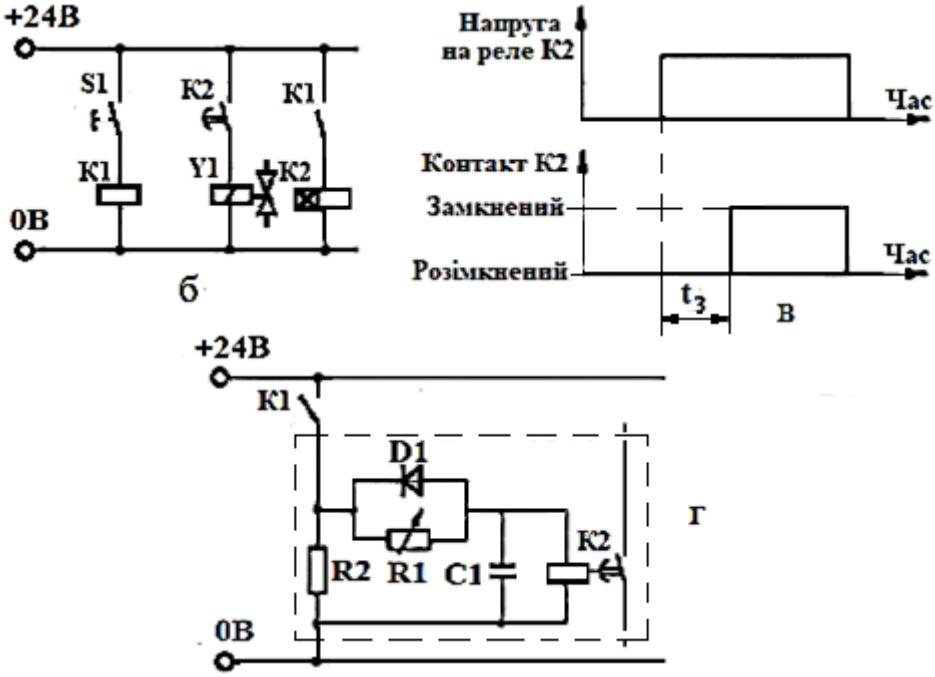


Рис. 7.11. Схема непрямого включення реле часу із затримкою включення котушки електромагніту пневморозподільника Y1: а – діаграма включення / вимикання світлодіоду із затримкою часу 500 мс; б – схема включення реле часу; в – діаграма роботи реле часу; г – електрична схема реле часу К2

У реле часу К2 на рис. 7.11 контакт К2 замикається без затримки часу при натисканні кнопки S1, а розмикається через заданий інтервал часу затримки $t_з$ після відпускання кнопки старту S1. а скидання станеться без затримки часу після відпускання кнопки S1.

У реле часу К2 на рис. 7.12 контакт К2 замикається через заданий інтервал часу *tc* після натискання кнопки старту S1 і спрацьовані реле К1 (подання сигналу), а скидання станеться без затримки часу після відпускання кнопки S1. Час затримки регулюється резистором R1 (рис. 7.12,а). Реле часу працює наступним чином. ЯКЩО натиснути кнопку S1, струм проходить через змінний резистор R1 і конденсатор C1 заряджається. Після зарядки конденсатора до певної ємкості реле К2 включається і його нормально розімкнений контакт К2 замикається в ланцюгу Y1. ЯКЩО кнопку S1 відпустити від натискання, ланцюг розімкнеться і конденсатор C1 швидко розряджається через діод D1 і резистор R2. В результаті реле часу практично миттєво повертається у початковий стан, тобто його контакт К2 розмикається.

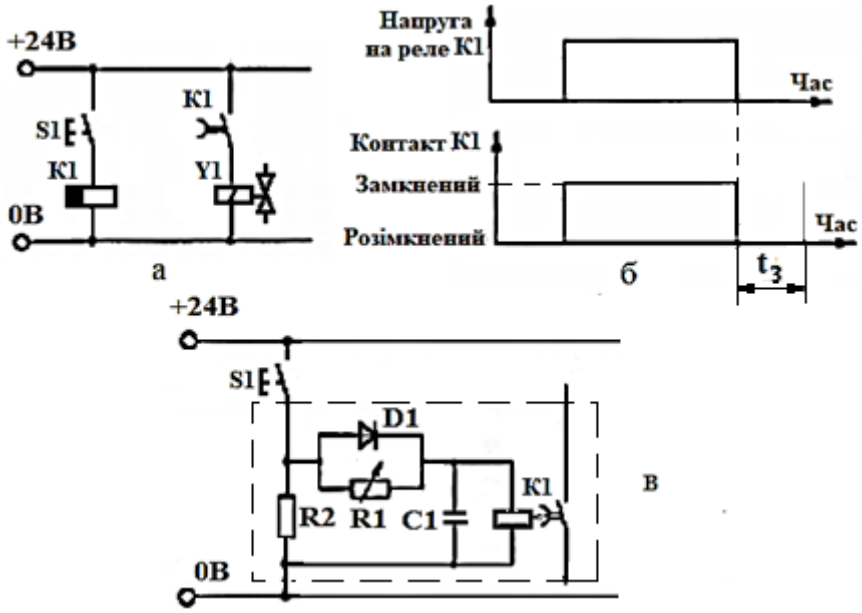


Рис. 7.12. Схема прямого включення реле часу із затримкою виключення котушки електромагніту пневморозподільника Y1: а – схема включення; б – діаграма роботи; в – електрична схема реле часу К1

ТАЙМЕР-ОБ’ЄКТ (рис. 7.13) також як і ЛІЧИЛЬНИК-ОБ’ЄКТ (рис. 7.10) у циклових системах керування з контролером застосовуються ні як фізичні електромеханічні або електропневматичні пристрої, а як програмні об’єкти контролера. В релейно-контактних схемах циклового керування без контролера ТАЙМЕРИ і ЛІЧИЛЬНИКИ, це фізичні елементи електричних схем мехатроніки.

На технологічної мові STL програмування ТАЙМЕР-ОБ’ЄКТу також як і ЛІЧИЛЬНИК-ОБ’ЄКТу складається з трьох частин (рис. 7.13), а саме одно бітового операнда стану ТАЙМЕРА (**T**n) і двох 16-бітових операндів: ПРЕСЕЛЕКТОРА ТАЙМЕРА (**TP**n) і СЛОВА ТАЙМЕРА (**TW**n).

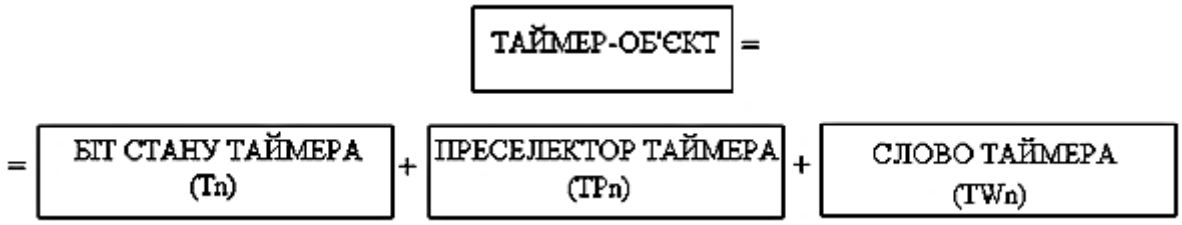


Рис. 7.13. Структурна схема ТАЙМЕРА як об’єкта на мові STL

В неактивний (вимкнений) стан (**T**n=0) біт стану ТАЙМЕРn переводиться командою RESET. Також в неактивний стан (**T**n=0) ТАЙМЕРn автоматично переводиться коли запрограмований період часу затримки таймера завершений. Як відрізнити ці два «нулі» на програмному рівні? Для цього вводяться програмні маркери (елементи пам'яті) під назвою ФЛАГn (**F**n) з поширеною назвою «*флаг покриття таймера*».

У зв'язку з цим надалі для двох складових ТАЙМЕРА-ОБ'ЄКТА будемо розрізняти активний «1» стан і неактивний (нулевий) стан «0» за допомогою наступних трьох ФЛАГІВ цільового призначення.

F0.0:=1/0 це ФЛАГ початкового стану ТАЙМЕРА (таймер увімкнений/вимкнений). Тобто, якщо ФЛАГ початкового стану ТАЙМЕРА (**F**0.0=1) приймають значення «1» це позначає що таймер включається і починає роботу за командою **SET**. В результаті утворюється БІТ активного стану ТАЙМЕРn (**T**n=1).

F0.1:=1/0 це ФЛАГ кінцевого стану ТАЙМЕРА (таймер відрахував час затримки/ще рахує час затримки).

F0.2 :=1/0 це ФЛАГ ПРЕСЕЛЕКТОРА ТАЙМЕРА (ФЛАГ ПОКРИТТЯ ТАЙМЕРА). Ініціалізація преселектора таймера і завантаження числа (часу затримки) відбувається командою **LOAD Vn TO TP**n (приклад 7.1 і приклад 7.2). ПРЕСЕЛЕКТОР ТАЙМЕРАn (**TP**n) служить для попередньої установки значенням (числом **V**) 16-бітового операнда **TP**n для таймера **T**n. При цьому ініціалізація преселектора таймера **TP**n може бути з часовим класом що заданий явно (приклад 7.1) і ініціалізація **TP**n без завдання часового класу (приклад 7.2).

СЛОВО ТАЙМЕРА n (**TW**n) - 16-бітовий операнд, в який при запуску ТАЙМЕРАn командою **SET** автоматично передається (копіюється) значення (числа **V**) ПРЕСЕЛЕКТОРА ТАЙМЕРАn (**TP**n). Зміст СЛОВА ТАЙМЕРА (**TW**n) автоматично зменшується контролером через сталі інтервали часу поки СЛОВО ТАЙМЕРА не буде дорівнюватися нулю (**TW**n =0).

Програмування таймера виконується в наступної послідовності: команда ІНІЦІАЛІЗАЦІЇ ПРЕСЕЛЕКТОРА (пре установка або покриття) ТАЙМЕРАn; команда на ЗАПУСК ТАЙМЕРАn; команда ПЕРЕВІРКИ СТАНУ ТАЙМЕРАn (працює/зупинений).

1. ІНІЦІАЛІЗАЦІЯ ПРЕСЕЛЕКТОРА ТАЙМЕРА

Приклад 7.1. Ініціалізація преселектора **TP1** таймера **1** ***з*** *часовим класом.*

```
STEP1
IF          NOP     'завжди виконання
    LOAD     V5     'ЗАВАНТАЖИТИ значення 5 затримки часу
TO          TP1     'в ПРЕСЕЛЕКТОР ТАЙМЕРА 1
    WITH    SEC     'з часовим класом «секунди»
```

В результаті цього фрагменту програми (ВИРАЗУ) Таймер **1** запрограмований на затримку часу, що складає 5 секунд. Можливі наступні часові класи для таймера:

HSC – (one **H**undred **SEC**) соті частки секунди;
TSC – (**T**en **SEC**) десяті частки секунди;
SEC – секунди;
MIN – мінути.

Приклад 7.2. Ініціалізація преселектора **TP1** таймера **1** *без часового класу*.

```
STEP1
IF          NOP     'завжди виконання
LOAD       V500     'значення 500 завантажити
TO          TP1     'в ПРЕСЕЛЕКТОР ТАЙМЕРА 1
```

В результаті цього фрагменту програми (виразу) невстановлений часовий клас буде складати 1/100 секунди і тому затримка часу таймера 1, яка задана у преселекторі **TP1** буде складати також 5 секунд (500*1/100= 5 с.). Діапазон часу може складати 0…65535, що забезпечує час затримки таймерів від 0,01 до 655,35 секунд.

2. ЗАПУСК ТАЙМЕРА

Запуск таймера виконується командою **SET**:

```
IF    I.1.0        'ЯКЩО вхід 1.0 активний (тут може бути
                   'будь-яка умова)
THEN SET  T1       'ТОДІ запустити Т1 - таймер 1
```

Після виконання команди **SET T1** (у загальному випадку команди **SET T**n) програма автоматично виконує наступні 4 дії:

1. Значення (**TP1**) ПРЕСЕЛЕКТОРА ТАЙМЕРА **1** копіюється в СЛОВО (**TW1**) ТАЙМЕРА **1** , тобто **TW1:= TP1**;

2. Стан ТАЙМЕРА **1** встановлюється в активний стан (**T1=1**) і ТАЙМЕР 1 запускається в роботу;

3. Значення в **TW1:= TP1** автоматично зменшується контролером у відповідності із визначеним інтервалом часу (*приклад* а або *приклад* б);

4. Коли СЛОВО ТАЙМЕРА **1** досягне «0» (**TW1=0**), ТОДІ статус ТАЙМЕРА встановлюється в «0» стан (**T1=0**), тобто таймер 1 спрацьовує із заданою ПРЕСЕЛЕКТОРОМ ТАЙМЕРА (**TP1**) затримкою часу.

3. ПЕРЕВІРКА СТАНУ ТАЙМЕРА

Перевірка активності таймера, наприклад **T1** на протязі часу затримки, який встановлений преселектором таймера виконується в умовній частині ВИРАЗУ:

```
IF    T1           'ЯКЩО таймер 1 включений (працює)
IF N  T1           'ЯКЩО таймер 1 вимкнений (зупинений)
```

Зупинення ТАЙМЕРА

Для зупинення таймера треба застосувати команду **RESET** і задати його номер n таймера:

```
IF       I1.0          'ЯКЩО вхід 1.0 включений (працює),
THEN RESET T1          'ТОДІ зупинити ТАЙМЕР 1
```

При цьому біт СТАНУ ТАЙМЕРА 1 встановлюється в стан (**T1=0**). ЯКЩО таймер 1 вже був в неактивному стані (**T1=0**), то нічого не відбувається.

При крокової структурі програми, ЯКЩО *умовна частина виразу* є «ІСТИНА», то кожній раз будуть виконуватися команди, які задані у *виконавчій частині* цього *виразу*. В мові STL умова істинності ВИРАЗУ оцінюється в кожному циклі програми без врахування попереднього стану цього ВИРАЗУ. Тому щоб уникнути неконтрольованого багатократного виконання більшості команд типу **SET T**n для таймерів, команд **INC/DEC** СЛОВО ЛІЧИЛЬНИКА (**CW**n), команд **SHL/SHR** та інших використовують ФЛАГІ, як засіб контролю стану (1/0).

Запобігання повторного включення ТАЙМЕРА **без операнда** ФЛАГ

В цьому прикладі розглянуто запрограмоване включення електродвигуна приводу **O1.0** на час 10 секунд (час прокрутки) за допомогою *таймера* **T1** та натисканні кнопки «Start» **I.1.0** при умові що включення відбувається після паузи 5 секунд, яка визначається *таймером* **T2** і для запобігання повторного включення (перезапуску) таймерів використовується поєднання команди **STEP**n з командою **SET T**n (запуск таймера n).

```
STEP 1
IF          NOP          'завжди виконувати (ініціалізація
                         'при включенні живлення)
THEN  LOAD  V500         'завантажити преселектор таймера 1 паузи
        TO  TP1          'числом 500 для часу паузи 500*.01=5 с
LOAD    V1000            'завантажити преселектор таймера 2
        TO  TP2          'числом 1000 для часу прокрутки 1 с
SET         T1           'увімкнути таймер T1 паузи

STEP 5                   'вираз 1 основної програми
IF      N   T1           'ЯКЩО таймер 1 відпрацював
   AND  N   T2           'і таймер 2 ні працює
   AND  N   O1.0         'і електродвигун O1.0 вимкнений
   AND      I.1.0        'і натиснута кнопка «Start» 1.0
THEN  SET   T2           'ТОДІ запустити таймер 2,
      SET   O1.0         'та увімкнути електродвигун 1.0

STEP 10                  'вираз 2 основної програми
```

```
IF   N   T2              'ЯКЩО час прокрутки електродвигуна
                         'закінчився(таймер 2 відпрацював 10  с)
THEN RESET  O1.0         'ТОДІ зупинити електродвигун 1.0
SET      T1              'та знову запустити таймер 1 паузи  5 с
  JMT TO 5               'ПЕРЕЙТИ на STEP 5, тобто почати
                          'роботу електродвигуна спочатку
```

Запобігання повторного включення (перезапуску) ТАЙМЕРА **з використанням операнда** ФЛАГ

В цьому прикладі **F1.0** (ФЛАГ 1.0) застосований як елемент пам'яті одноразового натискання кнопки «Start» 1.0 в циклової системі керування пневмоциліндром **O1.0**, наприклад з витримкою часу 2 с за допомогою **T0** (таймера 0) і **TP0** (преселектора таймера 0).

```
STEP1                    'однократна ініціалізація
THEN  LOAD    V0
      TO      OW0        'виключити всі ВИХОДИ
RESET         F1.0       'скинути флаг 1.0
      LOAD    V200       'завантажити число 200
           TO TP0        'у преселектор таймера 0 і цей таймер
                         'стає 2-секундним(200*0.01=2 с)
STEP 2                   'основна програма
IF          I1.0         'ЯКЩО кнопка Start 1.0 натиснута
AND     N     T0         'і таймер T0 ні працює
AND     N     F1.0       'і флаг F1.0 скинутий,
THEN  SET     T0         'ТОДІ запустити таймер 0,
      SET     O1.0       'увімкнути  пневмоциліндр 1.0
      SET     F1.0       'увімкнути флаг натиснутої кнопки 1.0
IF      N     T0         'ЯКЩО таймер 0 відпрацював
      AND     O1.0       'і пневмоциліндр 1.0 спрацював
THEN RESET    O1.0       'ТОДІ виключити пневмоциліндр 1.0
IF      N     T0         'ЯКЩО таймер 0 не активний
      AND     F1.0       'і флаг F1.0 пам'ятає одноразове
                         'натискання кнопки Start 1.0
      AND N   I1.0       'і кнопка Start 1.0 відпущена
```

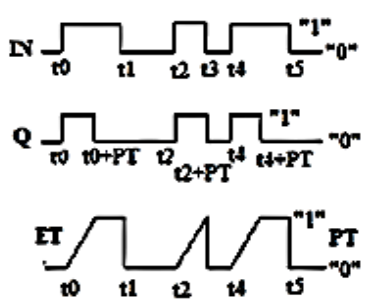
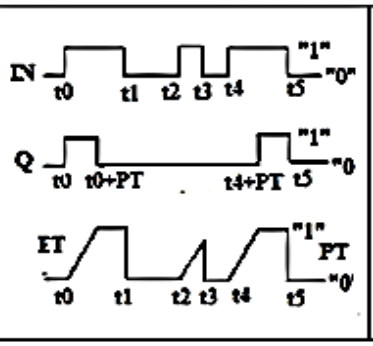
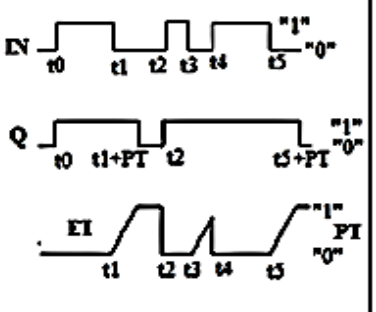
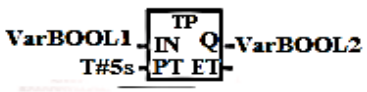
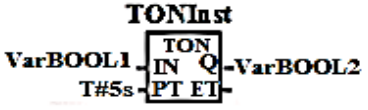
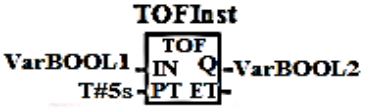
```
THEN RESET   F1.0      'ТОДІ  вимкнути флаг 1.0, тобто
                       'підготувати цей флаг до наступного
                       'натискання кнопки Start 1.0
IF           NOP       'виконувати завжди, якщо запрограмовані
                       'дії у виконавчій частині ВИРАЗУ
THEN JMT TO 2          'ТОДІ перейти на STEP 2, тобто
                       'продовжити цикл роботи з кроку STEP 2
```

Позначення функціонального блоку «Таймер» і приклади програмування таймеру на мовах IL, FBD і ST програмного середовища CoDeSys наведені в таблиці 7.2.

Наприклад, функціональний блок **TOF** «Таймер з затримкою вимикання» має позначення входів **IN** і **PT**, а виходів **Q** і **ET** на яких можуть бути сигнали типу **BOOL** := **TRUE** або **FALSE** і **TIME**:=**s** (секунди).

Якщо **IN** дорівнює **TRUE**, то вихід **Q** = **TRUE** і вихід **ET** = 0. Як тільки **IN** переходить в **FALSE**, починається відлік часу (в **мілісекундах**) на виході **ET**. При досягненні заданої тривалості відлік зупиняється. Вихід **Q** дорівнює **FALSE**, якщо **IN** дорівнює **FALSE** і **ET** дорівнює **PT**, інакше - **TRUE**. Таким чином, вихід **Q** скидається з затримкою **PT** від спаду входу **IN.**

Таблиця 7.2 Позначення і приклади програмування таймеру

Позначення функціонального блоку "Таймер" в різних режимах роботи		
TP **«таймер»** **TP(IN, PT, Q, ET)**	**TON** **«таймер з затримкою включення»** **TON(IN, PT, Q, ET)**	**TOF** **«таймер з затримкою виключення»** **TOF(IN, PT, Q, ET)**
Приклад оголошення: **TPInst : TP;**	*Приклад оголошення:* **TONInst: TON;**	*Приклад оголошення:* **TOFInst: TOF;**
Часові діаграми роботи таймеру в різних режимах		
		
Приклад програмування таймеру на мові **IL**		
CAL TPInst(IN := VarBOOL1, PT := T#5s) LD TPInst.Q ST VarBOOL2	CAL TONInst(IN := VarBOOL1, PT := T#5s) LD TONInst.Q ST VarBOOL2	CAL TOFInst(IN := VarBOOL1, PT := T#5s) LD TOFInst.Q ST VarBOOL2
Приклад програмування таймеру на мові **FBD**		
		
Приклад програмування таймеру на мові **ST**		
TPInst(IN := **VarBOOL1, PT:= T#5s);** **VarBOOL2 :=TPInst.Q;**	TONInst(IN := VarBOOL1,PT:= T#5s);	TOFInst(IN := VarBOOL1,PT:= T#5s); VarBOOL2 :=TOFInst.Q

Таймер це лічильник інтервалів часу. Скорочення: **TON** – **T**imer **ON** (таймер вимкнення); **TOF** – **T**imer **OF**F (таймер включення).

Розглянемо приклади побудови програм PT1…PT8 крокових структур і без крокових структур роботи таймеру в різних режимах. В таблиці 7.3 в Allocation List прописані типи операндів символьних імен операндів, і відповідні їм абсолютні імена та коментарі до цих операндів.

Таблиця 7.3 Allocation List для програмування таймерів

№	Тип операнда	Abs. і'мя в ПЛК	Коментар за змістом дії (не транслюється програмою)	
			Англомовний	Україномовний
1	Вихід	O0.0	Output flashes without steps	Вихід, що мигає програми без кроків для таймера
2	Вихід	O0.1	Output flashes with steps	Вихід, що мигає програми з кроками для таймера
3	Вихід	O0.2	Timer OFF delay without steps	Вихід, який спрацьовує при затримки часу вимикання таймера програми без кроків
4	Вихід	O0.3	Timer OFF delay with steps	Вихід, який спрацьовує при *затримки часу вимикання* таймера програми з кроками
5	Вихід	O0.4	Timer ON delay without steps	Вихід, який спрацьовує при *затримки часу вмикання* таймера у програмі без кроків
6	Вихід	O0.5	Timer ON delay with steps	вихід, який спрацьовує при *затримки часу вмикання таймера* програми з кроками
7	Вихід	O0.6	Output visua-lising the puls	Вихід, який показує пульсацію
8	Вихід	O0.7	Puls timer without steps	Вихід, який показує режим пульсації таймера програми з кроками
9	Вхід	I0.4	Start timer OFF delay	Вхід початку роботи таймера вимикання
10	Вхід	I0.5	Start timer ON delay	Вхід початку роботи таймера вмикання
11	Вхід	I0.6	Start timer	Вхід початку роботи таймера
12	Флаг	F0.0	Edge for Timer ON	Елемент пам'яті для таймера вмикання
13	Флаг	F0.1	Edge for Timer OFF	Елемент пам'яті для таймера вимикання
14	Таймер	T0	Puls timer using steps	Таймер пульсації програми з кроками
15	Таймер	T1	Puls timer without steps	Таймер пульсації програми без кроків

16	Таймер	T2	Timer ON delay using steps	Таймер вмикання програми з кроками
17	Таймер	T3	Timer ON delay without steps	Таймер вмикання програми без кроків
18	Таймер	T4	Timer OFF delay using steps	Таймер вимикання програми з кроками
19	Таймер	T5	Timer OFF delay without steps	Таймер вимикання програми без кроків
20	Таймер	T6	Timer Flashing time using steps	Таймер мерехтіння програми з кроками
21	Таймер	T7	Timer Flashing time without steps	Таймер мерехтіння програми без кроків
22	Програ-ма	PT1	Puls timer using steps	Програма пульсуючого таймера з кроками
23	Програ-ма	PT2	Puls timer without steps	Програма пульсуючого таймера без кроків
24	Програ-ма	PT3	Timer ON delay using steps	Програма таймера вмикання з кроками
25	Програ-ма	PT4	Timer ON delay without steps	Програма таймера вмикання без кроків
26	Програ-ма	PT5	Timer OFF delay using steps	Програма таймера вимикання з кроками
27	Програ-ма	PT6	Timer ON delay without steps	Програма таймера вимикання без кроків
28	Програ-ма	PT7	Flashing with steps	Програма миготіння з кроками
29	Програ-ма	PT8	Flashing without steps	Програма миготіння без кроків

При створенні проектів в середовищі FST мовою STL передбачається наступний порядок розробки програм. Спочатку потрібно прописати **Allocation List** (Алокейшен Лист), який встановлює відповідність *символьним іменам операндів* що використані у програмі їх *абсолютних іменам* , які прив'язані до входів/виходів (Input/Output) контролера та до ділянок пам'яті контролера для таймерів, лічильників, флагів (маркерів, елементів пам'яті). Програми можуть мати структуру з кроками (STEPs) або мати структуру без кроків.

З урахуванням Алокейшен Листа (табл. 7.3) розглянемо порядок розробки програми **PT0**, яка забезпечує 8 програм **PT1**…**PT8** початку роботи таймерів в різних режимах в схемах мехатроніки.

З а у в а ж е н н я: позначкою «‘» на початку кожного рядка в програмі тут і надалі позначені ***коментарі до програми***, які ігноруються при компіляції програми.

Програма PT0. Програма P0 забезпечує початок роботи програм **PT1**…**PT8** крокових структур і без крокових структур при відсутності команд (Start all programs PT1…PT8)

```
    IF  NOP          ‘ЯКЩО команда NOP ЗАВЖДИ
THEN SET PT1         ‘ТОДІ ЗАДАТИ програму PT1 з кроками
     SET PT2         ‘ЗАДАТИ програму PT2 без кроків для  таймера
     SET PT3         ‘ЗАДАТИ програму PT3 з кроками
     SET PT4         ‘ЗАДАТИ програму PT4 без кроків
     SET PT5         ‘ЗАДАТИ програму PT5 з кроками
     SET PT6         ‘ЗАДАТИ програму PT6 без кроків
     SET PT7         ‘ЗАДАТИ програму  миготіння PT7 з кроками
     SET PT8         ‘ЗАДАТИ програму миготіння PT8 без кроків
```

Програма PT1. Програма **PT1** (**крокової структури**) забезпечує вмикання і вимикання виходу **O0.6** на заданий час після сигналу **I0.6**. Ця програма забезпечує режим пульсації таймера.

```
STEP Init
IF          I0.6       ‘ЯКЩО є наявність сигналу I0.6
THEN  SET     T0       ‘ТОДІ увімкнути таймер T0
     WITH     5s       ‘на 5 секунд
STEP Check_Tim
IF           T0        ‘ЯКЩО увімкнений таймер T0
THEN   SET   O0.6      ‘ТОДІ увімкнути вихід O0.6
 OTHRW RESET O0.6      ‘В іншому випадку вимкнути вихід  O0.6
IF    N      T0        ‘При відпрацюванні таймера T0
     AND N   I0.6      ‘І відсутності сигналу I0.6
THEN  JMP TO Init      ‘Перейти до кроку STEP Init
```

```
IF          NOP              'ЯКЩО нічого не відбувається
THEN JMP TO Check_Tim        'ТОДІ перейти до кроку Check_Tim
```

Програма PT2. Програма **PT2** (**з структурою циклу без кроків**) також забезпечує вмикання і вимикання виходу **O0.6** на заданий час після сигналу **I0.6** і забезпечує також режим пульсації таймера.

```
STEP Init
IF          I0.6        'При наявності сигналу I0.6
   AND N    T1          'І вимкненому таймері T1(табл.7.3)
THEN  SET   T1          'Увімкнути таймер T1
     WITH   5s          'на 5 секунд

IF           T1         'При ввімкненому таймері T1
THEN  SET   O0.7        'Увімкнути вихід O0.7
OTHRW RESET O0.7        'В іншому випадку вимкнути O0.7
```

Програма PT3. Програма **PT3** (**крокової структури**) забезпечує вмикання виходу **O0.5** після затримки в часі при наявності сигналу **I0.5**. Ця програма таймера вмикання має крокову структуру і забезпечує режим роботи таймера із запізненням при вмиканні.

```
STEP Init
IF          I0.5            'При наявності сигналу I0.5
THEN  SET   T2              'Таймер T2 починає роботу
     WITH   5s              'на 5 секунд

STEP wait
IF     N    I0.5            'ЯКЩО сигналу I0.5 відсутній
THEN  RESET  T2             'Тоді вимкнути таймер T2
JMP TO Init                 'та перейти до кроку STEP Init
                  'Включення виходу після закінчення часу затримки
IF     N    T2              'ЯКЩО таймер T2 відпрацював
      AND   I0.5            'і вхід I0.5, як і раніше дорівнює «1»
THEN  SET   O0.5            'ТОДІ вихід O0.5 буде увімкнений і
        'надалі при відпрацюванні таймера T2 і наявності сигналу I0.5
```

```
STEP end
IF      N     I0.5          'При відсутності сигналу I0.5
THEN RESET   O0.5           'Вимкнути вихід O0.5
JMP TO Init                 'та перейти до кроку STEP Init
```

Програма PT4. Програма **PT4 (з структурою циклу без кроків)** забезпечує вмикання виходу **O0.4** після затримки в часі при наявності сигналу I0.5. Ця програма також забезпечує режим роботи таймера із запізненням при вмиканні як і програма **PT3**.

```
IF    N     T3            'ЯКЩО таймер T3 вимкнений
     AND    I0.5          'І є сигнал на вході I0.5
     AND N  F0.0          'І вимкнений флаг F0.0 (табл.7.3)
THEN SET    T3            'ТОДІ увімкнути в роботу таймер T3
     WITH   5s            'на 5 секунд затримки часу
      SET   F0.0          'та умикнути елемент пам'яті F0.0
'Таймер вимикається, як тільки на вході IO.5 з'являється «0»
IF     N       I0.5       'При відсутності сигналу I0.5
 THEN RESET    T3         'вимкнути таймер T3
       RESET   O0.4       'вимкнути вихід O0.4
       RESET   F0.0       'і вимкнути елемент пам'яті F0.0
'Вихід O0.4 вмикається, як тільки закінчується час витримки таймеру T3.
'При цьому вхід I0.5 залишається увімкненим
 IF    N     T3           'ЯКЩО таймер T3 відпрацював
      AND    F0.0         'і є сигнал елемента пам'яті F0.0
      AND    I0.5         'і вхід I0.5 залишається  увімкненим
 THEN  SET O0.4           'ТОДІ увімкнути вихід O0.4
```

Програма PT5. Програма **PT5 (крокової структури)** також вмикає вихід **O0.3** при наявності сигналу **I0.4** і вимикає його після зникнення сигналу **I0.4** через заданий час

```
STEP Init
IF          I0.4          'При наявності сигналу I0.4
THEN  SET O0.3            'Вмикання виходу O0.3
```

```
IF     N    I0.4        'ЯКЩО відсутній сигнал I0.4
       AND  O0.3        'і ввімкнений O0.3
     THEN   SET T4      'ТОДІ  увімкнути таймер T4
       WITH   5s        'на 5 секунд

STEP Wait
IF     N       T4       'При відпрацюванні таймера T4
THEN   RESET   O0.3      'вимкнути вихід  O0.3
     JMP TO Init         'і перейти до кроку Init
IF          I0.4        'ЯКЩО є сигнал на вході I0.4
THEN   JMP TO Init       'ТОДІ перейти до кроку Init
```

Програма PT6. Програма **PT6** (**з структурою циклу без кроків**) також вмикає вихід **O0.3** при наявності сигналу **I0.4** і вимикає його після зникнення сигналу **I0.4** через заданий час.

```
'При наявності сигналу I0.4 - увімкнути вихід O0.2 і вимкнути  елемент
'пам'яті F0.1
IF           I0.4
THEN   SET   O0.2
     RESET   F0.1
'При відсутності сигналу I0.4 і ввімкненому виході O0.2 і вимкнутому
'елементі пам'яті  F0.1 – увімкнути таймер T5 на 5 секунд і увімкнути
'елемент пам'яті F0.1
IF      N      I0.4
        AND    O0.2
        AND N  F0.1
THEN    SET    T5
        WITH   5s
SET            F0.1
'При відпрацюванні таймера T5 і наявності сигналу від елемента пам'яті
'F0.1 – вимкнути вихід O0.2
IF      N      T5
        AND    F0.1
THEN RESET     O0.2
```

Програма PT7. Програма **PT7 (крокової структури)** забезпечує періодичне вмикання/вимикання виходу **O0.1** за допомогою крокової структури програми

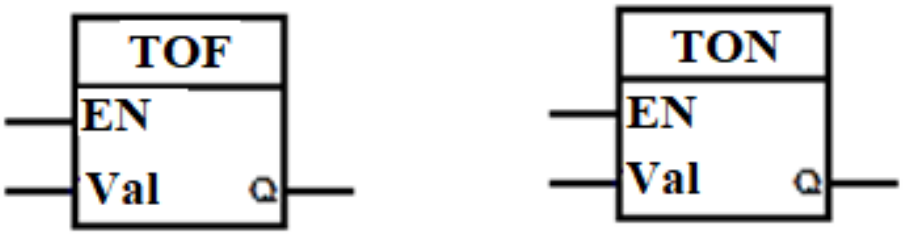
```
STEP Init
'При відсутності команд – увімкнути таймер T6 на 1 секунду і увімкнути
'вихід O0.1
IF            NOP
THEN  SET     T6
      WITH    1s
      SET     O0.1

STEP Wait
'Після відпрацювання таймера T6 вимкнути O0.1 і увімкнути таймер T6
 IF     N     T6
 THEN RESET O0.1
 SET T6
 STEP End

'ЯКЩО таймер T6 відпрацював – перейти до кроку Init
 IF     N       T6
 THEN   JMP TO Init
```

Програма PT8. Програма **PT8 (з структурою циклу без кроків)** служить для періодичного вмикання/вимикання виходу **O0.0** програми.

```
'Таймер запускає себе постійно при вимкненому таймері T7, а саме
'ЯКЩО  таймер T7 вимкнений ТОДІ увімкнути таймер T7 на 1 секунду і
'призначити виходу O0.0 значення протилежне дійсному значення
IF       N     T7
THEN   SET     T7
  WITH         1s
LOAD     N    O0.0
  TO          O0.0
```

7.5. **Програмування таймерів для платформи Arduino в середовищі FLProg з використанням графічної мови програмування FBD**

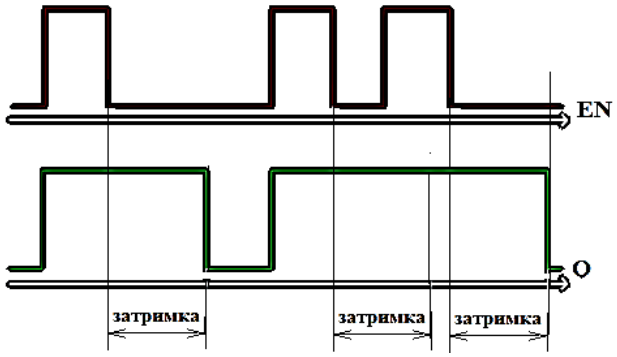
В програмному середовищі **FLProg** на графічної мові **FBD** (**F**unctional **B**lock **D**iagram стандарту МЕК 61131-3) таймер (блок **Таймери** – рис. 4.26,б розділу 4) це програмні блоки програми, які наведені на рис. 7.14.



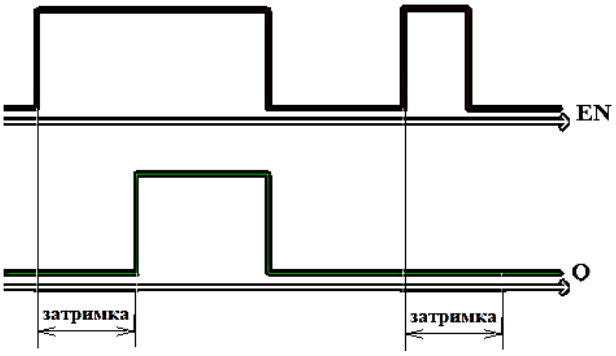
а б

Рис. 7.14. Програмні блоки **програми FLProg**: а – таймер типу **TOF** (**T**imer **Off**-delay) ***із затримкою включення***, який формує затримку часу по переднього фронту імпульсу (рис. 7.15,а), що подаються на вхід «EN» (від. **En**able - увімкнути); б – таймер типу **TON** (**T**imer **On**-delay) ***із затримкою вимкнення***, який формує затримку часу по заднього фронту імпульсу (рис. 7.15,б), що також подаються на вхід «EN»

На рис. 7.15 наведені типові тимчасові діаграми роботи таймерів типу TOF (із затримкою на відключення) та таймерів типу TON (із затримкою на включення).



а



б

Рис. 7.15. Вигляд тимчасових діаграм роботи таймерів: а – типу **TOF** (із затримкою на відключення); б – типу **TON** (із затримкою на включення)

Таймер на схемі в графічному середовищі **FLPprog** створюється шляхом перетягування блоку **Timer** (таймер) з бібліотеки програмних блоків (рис. 7.16).

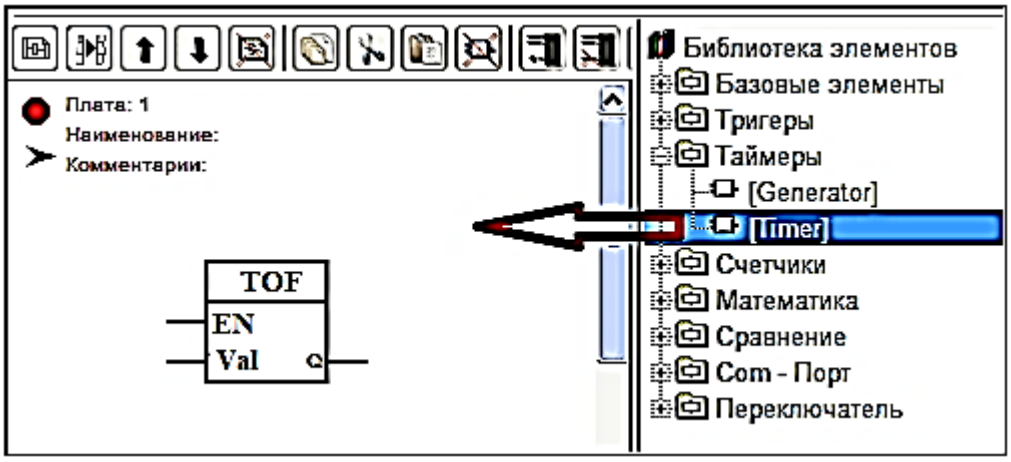


Рис. 7.16. Приклад перетягування блоку Timer з бібліотеки програмних блоків

Параметризація таймера виконується за допомогою редактора блоку Timer (рис. 7.17).

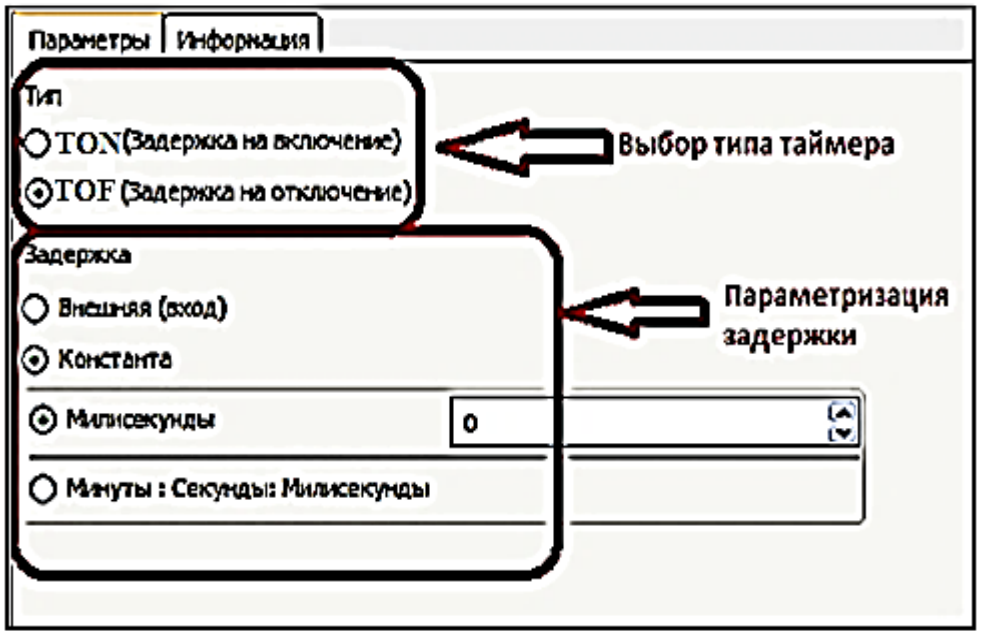


Рис. 7.17. Вікно редактора програмного блоку параметрів таймера

Величина затримки може бути задана у вигляді константи (7.17) або подана на вхід "Val" програмного блоку (7.16).

7.6. **Програмування лічильників мовою STL мехатронних систем технологічних машин та верстатів**

Практичне застосування датчиків кута повороту в мехатронних системах машин легкої промисловості розглянемо на прикладі програмування кількості стібків за допомогою ***лічильників*** в прямострочних автоматизованих швейних машинах з вбудованим контролером. На всіх швейних машинах 1 стібок виконується за 1 оборот головного валу машини. Тому датчик кута повороту головного валу після виконання кожного стібка надсилає на відповідних вихід (Output) контролера одну логічну «1». Але машинні стібки потрібно рахувати (додавати) при русі матеріалу «вперед» (від оператора) і віднімати при русі матеріалу «назад» (до оператора). При цьому використовується два виконавчих механізми з пневмоприводом або електромагнітним приводом.

Для виконання закріпок важіль реверсу матеріалу з пневмоприводом при виконанні одинарних закріпок і подвійних закріпок на початку і в кінці шва. Для автоматичної обрізки ниток використовується механізм обрізки ниток в кінці шва запрограмованої довжини після виконання

закріпки в кінці шва. Програмування лічильників також застосовується в циклових швейних машинах-автоматах для виготовлення петель і закріпок на одязі, машинах-автоматах для пришивання ґудзиків, плоских і круглих в'язальних автоматах, основов'язальних машинах та інших машинах легкої промисловості.

Позначення функціонального блоку «Лічильник» і приклади програмування лічильників на мовах **IL** і **FBD** програмного середовища **CoDeSys** (**Co**ntrollers **De**velopment **Sys**tem) наведені в таблиці 7.4. Інкрементний (increment – приріст) лічильник (поширена назва – *лічильник додавання*) додає на кожному кроці «1» після включення лічильника, а декрементний лічильник (поширена назва – *лічильник віднімання*) зменшує число на кожному кроці на «1» після команди **LOAD** (завантажити) числа в преселектор **CPn** лічильника (рис. 7.18).

Інкрементні лічильники та декрементні лічильники утворюються програмно командами **CTU** та **CTD**. Тобто оголошення типу лічильника в програмах на технологічних мовах програмування **FBD** та **IL** стандарту **IEC-61131**, наведених в таблиці 7.4 відбувається наступними англомовними скороченнями:

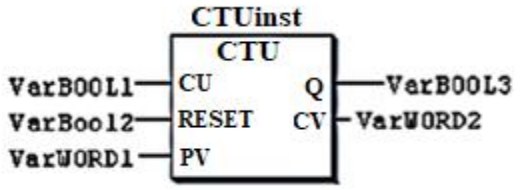
CTU – лічильник додавання (**прямий** або інкрементний лічильник);

CTD – лічильник віднімання (**зворотний** або декрементний лічильник).

У блоків синхронних лічильників (табл. 7.4) є наступні параметри:

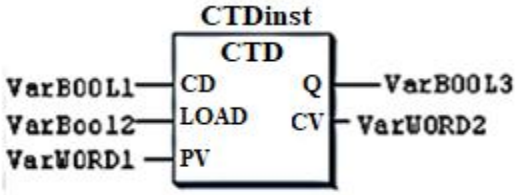
Таблиця 7.4. Позначення і приклади програмування лічильника на мовах **IL** і **FBD** програмного середовища CoDeSys

Інкрементний лічильник (л і ч и л ь н и к д о д а в а н н я)

Приклад на мові IL (Instruction List)	*Приклад на мові FBD (Functional Block Diagram)*
CAL CTDinst(CD:= VarBOOL1, LOAD:= VarBOOL2, PV:= VarWORD1, CV=> VarWORD2) LD CTDinst.Q ST VarBOOL3	

Входи: **CU** : BOOL; по переднему фронту на этом входе CV увеличивается на 1
RESET : BOOL; якщо TRUE, то CV сбрасивается в 0
PV : WORD; верхний предел значений CV
Виходи: **Q** : BOOL; стає TRUE, когда счетчик достигнет значения заданного PV
CV : WORD; значення, яке увеличивается на 1, пока не достигнет PV
Приклад оголошення: CTUInst : CTU ;

Декрементний лічильник (л і ч и л ь н и к в і д н і м а н н я)

Приклад на мові IL (Instruction List)	*Приклад на мові FBD (Functional Block Diagram)*
CAL CTDinst(CD:= VarBOOL1, LOAD:= VarBOOL2, PV:= VarWORD1, CV=> VarWORD2) LD CTDinst.Q ST VarBOOL3	

Входи: **CD** : BOOL; по переднему фронту на этом входе CV зменьшується на 1
LOAD : BOOL; якщо TRUE, CV сбрасивается в задане значення верхнього пределу (PV)
PV : WORD; верхній предел, т.е. начальное значення декремента.
Виходи: **Q** : BOOL; стає TRUE, когда CV стає 0
CV : WORD; значення, яке зменьшується на 1, починає з PV і до 0
Приклад оголошення: CTDInst : CTD ;

RESET – вхід скидання, при установці якого поточне значення лічильника **CV** (**C**urrent **V**alue – поточне значення) скидається в 0;

CU (**C**ount **U**p) – вхід прямого рахунку, що дозволяє збільшення CV на 1 при появі фронту CU = 0/1;

CD (**C**ount **D**own) – вхід зворотного рахунку, що дозволяє зменшення CV на 1 при появі фронту CD = 0/1;

PV (**P**review **V**alue) – попередньо встановлене значення лічильника;

LD (**L**oad **D**own) – завантаження початку зворотного відліку по фронту LD = 0/1.

На технологічної програмній мові STL лічильник-об’єкт складається *з трьох частин* (рис.7.18), а саме однобітового операнда **C**n – біта стану **Л**ічильника і двох МБ-операндів (16-бітових): **П**реселектора **Лі**чильника **CP**n і **С**лова **Л**ічильника **CW**n.

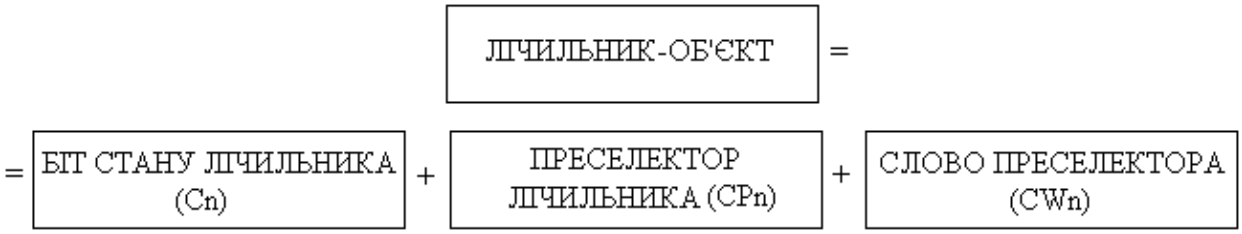


Рис. 7.18. Структурна схема ЛІЧИЛЬНИКА як об’єкта на мові STL

При створенні проектів з лічильником в середовищі FST мовою STL передбачається наступний порядок розробки програм. Спочатку потрібно прописати Allocation List (Алокешен Лист – таблиця 7.5) , який встановлює відповідність *символьним іменам операндів* що використані у програмі їх *абсолютних іменам* , які прив’язані до входів/виходів (Input/Output) контролера та до ділянок пам’яті контролера для таймерів, лічильників, флагів (маркерів, елементів пам’яті). Програми можуть мати структуру з кроками (STEPs) або мати структуру без кроків.

Таблиця 7.5 **Allocation List** для програмування лічильників

№ п/п	**Тип операнда**	**Abs. і’мя** в ПЛК	**К о м е н т а р** за змістом дії (не транслюється програмою для машинних кодів контролера)	
			Англомовний	Україномовний
1	Вихід	**O0.0**	**`Result forward counter using program steps`**	**Результат інкрементного лічильника програми з кроками**, а саме вихід контролера, який спрацьовує при відпрацюванні ***лічильника додавання*** у програмі **PC1**
2	Вихід	**O0.1**	**`Result forward counter without program steps`**	**Результат інкрементного лічильника без кроків програми,** а саме вихід контролера, який спрацьовує при відпрацюванні

				лічильника додавання у програмі **PC2**
3	Вихід	**O0.2**	**Result reverse counter using steps**	**Результат декрементного лічильника програми з кроками ,** а саме вихід контролера, який спрацьовує при відпрацюванні ***лічильника віднімання*** з кроками у програмі **PC3**
4	Вихід	**O0.3**	**Result reverse counter without steps**	**Результат декрементного лічильника програми без кроків,** а саме вихід контролера, який спрацьовує при відпрацюванні ***лічильника віднімання*** у програмі **PC4**
5	Вхід	**I0.0**	**Input signal for forward counter**	Вхідний сигнал для лічильника додавання
6	Вхід	**I0.1**	**Reset forward counter**	*Вимкнути* лічильник додавання
7	Вхід	**I0.2**	**Input signal for reverse counter**	Вхідний сигнал для лічильника віднімання
8	Вхід	**I0.3**	**Reset reverse counter**	Вимкнути лічильник віднімання
9	Флаг	**F0.0**	**Edge detection forward counter**	*Елемент пам'яті* лічильника додавання
10	Флаг	**F0.1**	**Edge detection reverse counter**	*Елемент пам'яті* лічильника віднімання
11	Програма	**PC0**	**Start all programs**	Початок роботи всіх програм лічильників
12	Програма	**PC1**	**Forward counter using steps**	Програма з кроками для ***прямого лічильника додавання*** **C0**
13	Програма	**PC2**	**Forward counter without steps**	Програма без кроків для ***лічильника додавання*** **C1**
14	Програма	**PC3**	**Reverse counter using steps**	Програма з кроками для ***лічильника віднімання*** **C2**
15	Програма	**PC4**	**Reverse counter without steps**	Програма без кроків для ***лічильника віднімання*** **C3**
16	Флаг	**FI**	**Init**	**В цьому**
17	Лічильник (counter) **0**	**C0**	**Forward counter with steps**	***Лічильник*** **C0** ***додавання*** для програми **P1**
18	Лічильник (counter) **1**	**C1**	**Forward counter without steps**	***Лічильник*** **C1** ***додавання*** для

				програми **P2**
19	Лічильник (counter) **2**	**C2**	**Reverse counter using steps**	***Лічильник* C2 *віднімання*** для програми **P3**
20	Лічильник (counter) **3**	**C3**	**Reverse counter without steps**	***Лічильник* C3 *віднімання*** для програми **P4**
21	Преселектор 16 біт лічильника **0**	**CP0**	**Nominal value counter 0**	Необхідне значення для *преселектора* **CP0** *лічильника* **C0**
22	Преселектор 16 біт лічильника **1**	**CP1**	**Nominal value counter 1**	Необхідне значення для *преселектора* **CP1** *лічильника* **C1**
23	Преселектор 16 біт лічильника **2**	**CP2**	**Nominal value counter 2**	Необхідне значення для *преселектора* **CP2** *лічильника* **C2**
24	Преселектор 16 біт лічильника **3**	**CP3**	**Nominal value counter 3**	Необхідне значення для *преселектора* **CP3** *лічильника* **C3**
25	Слово(**W**ord) 16 біт лічильника **2**	**CW2**	**Actual value counter 2**	Дійсне значення для *слова* **CW2** *лічильника* **C2**
26	Слово(**W**ord) 16 біт лічильника **3**	**CW3**	**Actual value counter 3**	Дійсне значення для *слова* **CW3** *лічильника* **C3**

З урахуванням Алокейшен листа (табл. 7.5) розглянемо приклади розробки програми **PC0**, яка забезпечує початок роботи наступних чотирьох програм **: програми PC1** та **програми PC3** (Steps-структури) для *лічильника додавання* (інкрементний лічильник) та *лічильника віднімання* (декрементний лічильник); **програми PC2** та **програми PC4** зі структурою без кроків для *лічильника додавання* та *лічильника віднімання*

З а у в а ж е н н я: позначкою « **'** » на початку кожного рядка в програмі тут і надалі позначені ***коментарі до програми***, які ігноруються транслятором.

```
'Початок роботи програми P0 увімкнути всі підпрограми P1…P4
IF          NOP   'ЯКЩО нічого не відбувається ТОДІ
THEN   SET  PC1   'почати програму  PC1 для лічильника додавання  C0
       SET  PC2   'почати програму  PC2 для лічильника віднімання C1
       SET  PC3   'почати програму  PC3 для лічильника додавання  C2
       SET  PC4   'почати програму  PC4 для лічильника віднімання C3
```

Програма PC1 – програма з кроковою структурою для підрахунку, наприклад, 25 машинних стібків при переміщенні матеріалу «вперед» та

виконанні шва заданої довжини на швейної машині з програмним керуванням. Ця програма вмикає вихід **O0.0** після виконання 25 оборотів головного валу швейної машини. Датчик кута поворотів можна імітувати 25-разовим натисканням кнопки S1, яка підключена до входу **I0.0** контролера на лабораторному стенді (рис. 9.1, розділ 9). Структура програми **PC1** побудована з використанням 4х кроків **STEP Init, STEP Count, STEP Continue** і **STEP Reset,** пояснення до яких і пояснення до команд наведені в коментарях (після позначок « ' », які ігноруються при компіляції програми).

```
STEP Init
 IF                NOP     'ЯКЩО нічого не відбувається
 THEN  LOAD        V25     'ТОДІ  завантажити число 25 (стібків)
         TO        CP0     'для преселектора CP0 лічильника C0
        SET         C0     'увімкнути лічильник C0

 'Рахунок і перевірка сигналу скидання лічильника

STEP Count
 IF        I0.1            'ЯКЩО є сигнал на вході I0.1
 THEN JMP TO Init          'ТОДІ повернутись до STEP Init
 IF        I0.0            'ЯКЩО є сигнал на вході I0.0
 THEN INC  C0              'ТОДІ ДОДАТИ «1» лічильнику C0

STEP Continue              'Оцінка стану лічильника
 IF         I0.1           'ЯКЩО є сигнал на вході I0.1
 THEN  RESET  C0           'ТОДІ вимкнути лічильник C0
     JMP TO Init           'та повернутись до STEP Init

 IF   N     I0.0           'ЯКЩО відсутній сигнал I0.0
    AND     C0             'і лічильник C0 ввімкнений
 THEN JMP TO Count         'ТОДІ перейти до STEP Count

 IF      N  I0.0           'ЯКЩО відсутній сигнал I0.0
    AND  N  C0             'і лічильник C0 вимкнений
 THEN SET   O0.0           'ТОДІ увімкнути вихід O0.0
```

```
STEP Reset                    'Вимкнення STEP
 IF          I0.1             'ЯКЩО є сигнал I0.1
 THEN RESET O0.0              'ТОДІ вимкнути вихід O0.0
      JMP TO Init             'повернутись до STEP Init

 IF           NOP             'ЯКЩО нічого не відбувається
 THEN JMP TO Reset            'ТОДІ повернутись до STEP Reset
```

Програма PC2 – програма за призначенням є функціонально адекватною програмі **PC1** але побудована зі структурою без кроків. Програма вмикає вихід **O0.1** після 15-разового вмикання входу **I0.0**

```
'Необхідне значення числа в лічильнику C1 встановлюється в
'преселекторі CP1 лічильника, наприклад число 15 відповідає, наприклад,
'15 запрограмованим стібкам ниткового шва після старту програми або
'при 15-разовому вмиканні входу (Input) I0.1 контролера

 IF           FI          'ЯКЩО ввімкнене флаг FI
     OR      I0.1         'АБО (OR) ввімкнене вхід I0.1
 THEN LOAD   V15     'ТОДІ ЗАВАНТАЖИТИ (THEN LOAD) число 15
     TO       CP1         'в преселектор CP1 лічильника
     SET       C1         'увімкнути лічильник C1
   RESET      O0.1         'та вимкнути вихід O0.1

IF           I0.0          'ЯКЩО є сигнал на вході I0.0
    AND N    F0.0          'і вимкнений елемент пам'яті F0.0
 THEN SET    F0.0     'ТОДІ увімкнути (THEN SET) елемент пам'яті
       INC            'F0.0  та додати (INC) одну одиницю до C1

'Вимкнення елемента пам'яті F0.0 лічильника  додавання
IF   N       I0.0       'ЯКЩО відсутній сигнал I0.0
THEN RESET F0.0     'ТОДІ вимкнути елемент пам'яті  F0.0
'Оцінка стану лічильника
IF   N         C1       'ЯКЩО лічильник C1 відпрацював
```

```
THEN  SET  O0.1      'ТОДІ увімкнути вихід (Output) O0.1
```

Програма PC3 – програма з кроковою структурою для підрахунку, наприклад 5 машинних стібків при переміщенні матеріалу «назад» для виконання закріпок на початку і в кінці шва. Ця програма вмикає вихід **O0.2** після виконання 5 оборотів головного валу швейної машини. Датчик кута поворотів можна імітувати 5-разовим натисканням кнопки S1, яка підключена до входу **I0.2** контролера на лабораторному стенді (рис. 9.1). Структура програми **P1** побудована з використанням кроків **STEP Init, STEP Count STEP Continue** і **STEP Reset,** пояснення до яких і пояснення до команд наведені в коментарях (після позначок « ' », які ігноруються транслятором).

```
'На кроці STEP Init відбувається завантаження необхідного значення
'лічильника 2

STEP Init
IF          NOP              'ЯКЩО нічого не відбувається
 THEN LOAD  V5               'ТОДІ ЗАВАНТАЖИТИ (LOAD) число 5
      TO    CW2              'для слова CW2 лічильника 2

'На кроці STEP Count відбувається рахунок і перевірка сигналу
'вимикання

STEP Count
 IF        I0.3              'ЯКЩО є сигнал на вході I0.3
 THEN  JMP TO Init           'ТОДІ перейти до кроку STEP Init

 IF        I0.2              'ЯКЩО є сигнал I0.2
 THEN  DEC  C2               'ТОДІ відняти(DEC) «1» у лічильника C2

'На кроці STEP Continue відбувається оцінка стану лічильника 2

STEP Continue
 IF        I0.3              'ЯКЩО є сигнал I0.3
 THEN RESET C2               'ТОДІ вимкнути лічильник C2
```

```
   JMP TO Init            'та перейти до кроку STEP Init

IF   N   I0.2             'ЯКЩО відсутній сигнал I0.2
    AND     C2            'і  увімкнений лічильник C2
 THEN JMP TO Count        'ТОДІ перейти до кроку STEP Count


IF   N   I0.2             'ЯКЩО відсутній сигнал I0.2
    AND N   C2            'і вимкнений лічильник C2
 THEN SET O0.2            'ТОДІ увімкнути вихід O0.2
```

'На кроці **STEP Reset** відбувається вимкнення виходу O0.2

```
STEP Reset
IF          I0.3          'ЯКЩО є сигнал на вході I0.3
THEN RESET O0.2           'ТОДІ вимкнути вихід O0.2
    JMP TO Init           'та перейти до кроку STEP Init

IF           NOP          'ЯКЩО нічого не відбувається
THEN JMP TO Reset         'ТОДІ перейти до кроку STEP Reset
```

Програма PC4 – програма за призначенням є функціонально адекватною програмі **PC3** але побудована зі структурою без кроків. Програма вмикає вихід **O0.3** після відпрацювання лічильника **C3.** При цьому слово **CW3** лічильника зменшується на «1» при кожному вмиканні входу **I0.1** командою **DEC**. Тобто після 5-разового вмикання входу (5 оборотів головного валу швейної машини) лічильник обнуляється і привод важелю реверсу на виході **O0.3** повертає важіль у верхнє положення.

```
IF          FI            'ЯКЩО увімкнений STEP флаг FI (Init)
      OR  I0.3            'АБО вхід I0.3 лічильника віднімання
THEN  LOAD  V5            'ТОДІ завантажити число 5(5 стібків)
      TO    CW3           'ДЛЯ слова CW3 лічильника 3
   RESET  O0.3            'та вимкнути вихід O0.3
IF           I0.2         'ЯКЩО є сигнал I0.2
  AND  N     F0.1         'і вимкнений елемент пам'яті F0.1
```

```
THEN   SET    F0.1     'ТОДІ увімкнути елемент пам'яті F0.1
     DEC        C3     'та відняти одну «1» від лічильника C3
 'вимкнення елемента пам'яті  F0.1 лічильника віднімання
IF       N     I0.2    'ЯКЩО відсутній сигнал I0.2
THEN   RESET F0.1      'ТОДІ вимкнути елемент пам'яті F0.1
 'Оцінка стану лічильника C3
IF       N       C3    'ЯКЩО лічильник C3 відпрацював
THEN    SET   O0.3     'ТОДІ увімкнути вихід O0.3
```

7.7. **Програмування лічильників для платформи Arduino в середовищі FLProg з використанням графічної мови програмування FBD**

На програмної мові **FBD** (**F**unctional **B**lock **D**iagram) стандарту МЕК 61131-3) лічильник (Counter) це програмний блок (програма), який підраховує імпульси, що подаються на вхід "C" (рис. 7.19).

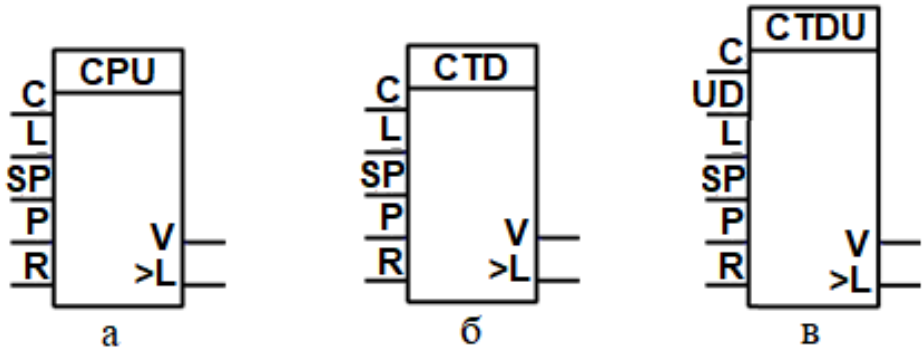


Рис. 7.19. Умовні позначення лічильників: а – **CTU** лічильник на збільшення – при приході переднього фронту імпульсу, значення лічильника збільшується на 1; б – **CTD** лічильник на зменшення – при приході переднього фронту імпульсу, значення лічильника зменшується на 1; в – **CTDU** лічильник на збільшення або на зменшення – напрямок підрахунку імпульсів залежить від наявності сигналу на вході "**UD**". При наявності сигналу лічильник рахує на збільшення, при відсутності сигналу лічильник рахує на зменшення.

Вхід **R** (**R**eset) служить для скидання лічильника. При наявності сигналу на цьому вході, значення лічильника скидається в 0.

Лічильник в графічному середовищі FLPprog створюється на схемі шляхом перетягування з бібліотеки елементів на робоче поле (рис. 7.20).

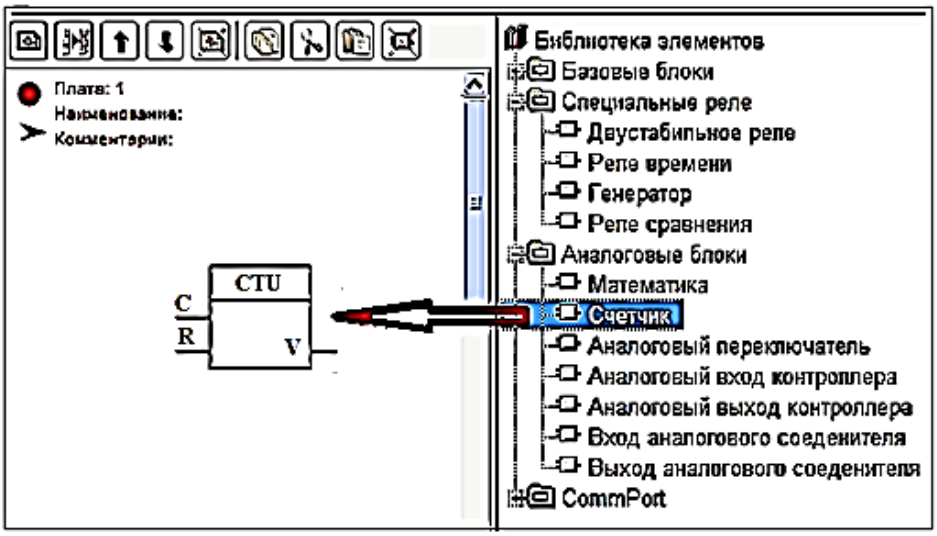


Рис. 7.20. Приклад перетягування блоку Counter з бібліотеки блоків

Параметри лічильника встановлюються за допомогою редактора програмного блоку (рис. 7.21).

Рис. 7.21. Вікно редактора програмного блоку параметрів лічильника

При включенні опції «**предустановка**» (попередня установка), у лічильника з'являється вхід **SP**. При наявності на цьому вході сигналу, в лічильник записується значення. Це значення може бути виставлено як константа або подаватися через вхід **SP** (рис. 7.19).

Рис.7.22. Вікно активного стану опції «Попередня установка» параметрів входу лічильника

Виходів у лічильника може бути один або два:

1. Вихід (**V)** (рис.7.19) це аналоговий вихід, на якому формується аналоговий сигнал, що відповідає поточному значенню лічильника.

2. Вихід (**> L)** це цифровий вихід, на якому видається сигнал, якщо значення лічильника більше або дорівнює певній величини. Ця величина може бути задана як константа або подана на вхід **L** (рис. 7.19).

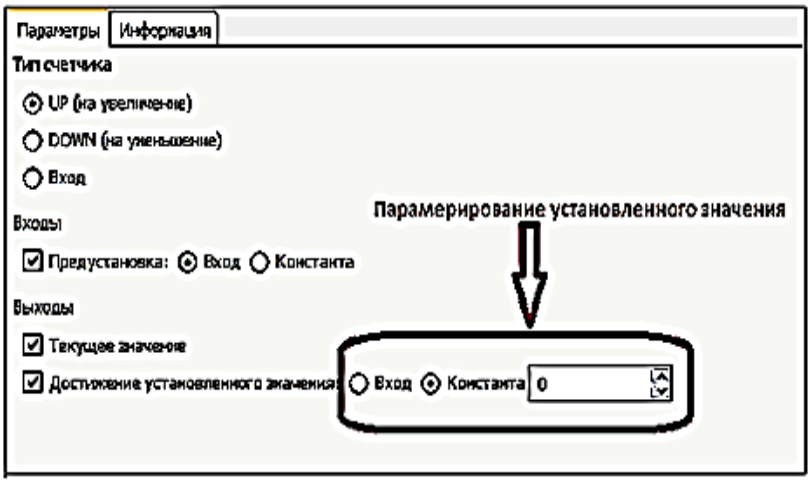


Рис. 7.23. Вікно активного стану опції «Досягнення встановленого значення» параметрів виходу лічильника

7.8. **Програмування ультразвукового датчика дальності на платформі Arduino**

Ультразвуковий датчик надсилає у простір УЗ-імпульси частотою 40 кГц (40 000 Гц) і заміряє час до повернення у приймач датчика першого ехо-сигналу. Знаючи цей час і швидкість звуку (340 м/с) програмно визначається відстань до об'єкту або дальність об'єкту. Після прийому першого ехо-сигналу приймач знову вмикається і регіструє наступні ехо-сигнали (рис. 7.24).

Принцип роботи датчика в термінах програми можна сформулювати наступним чином :

1.Подаємо імпульс (Start Pulse) тривалістю 10 мікросекунд на вхід **Trig** датчика. Вхід **Trig** з'єднаний з цифровими піном 9 плати контролера (рис. 7.24,д і рис. 7.25). Цей стартовий імпульс призначений для запуска циклу роботи мікросхеми контролера датчика:

```
digitalWrite(Trig,HIGH);  //запис імпульсу на вході Trig датчика
delayMicroseconds(10);    //тривалістю 10 мкс
```

2. Мікросхема контролера датчика за стартовим імпульсом генерує 8 імпульсів з частотою ультразвуку 40 кГц. Період однієї хвилі ультразвуку дорівнюється 1с/40 000 = 25 мкс. Тривалість 8 ультразвукових хвиль складає 8*25 мкс = 200 мкс (рис. 7.24,д). Така порція ультразвукових хвиль має назву **click** – рис. 7.1,а (*укр*. клацання, *рос.* щелчок). Цей **click** тривалістю 200 мікросекунд надсилається через «**Т-око**» датчика до об'єкту.

3. Дійшовши до об'єкту (перешкоди), **click** відбивається і **Echo**-сигнал приймається «**R-оком**» датчика. Отримуємо вихідний сигнал на виводі **Echo**, який з'єднаний з цифровими піном 8 плати контролера (рис. 7.24,д і рис. 7.25):

```
impulseTime=pulseIn(Echo,HIGH);   // заміряємо довжину
                                  // імпульсу Echo-сигналу
distance_sm=impulseTime/58;       // перераховуємо довжину
                                  // імпульсу в сантиметри
```

4. Безпосередньо на стороні контролера переводимо отриманий сигнал у відстань за формулою (7.6).

5. Для роботи з УЗД в програмах для платформи **Arduino**, платформи **Propeller Activity Board** для робототехніки та інших використовується бібліотека **Ping**. Назва **Ping** утворена від першої

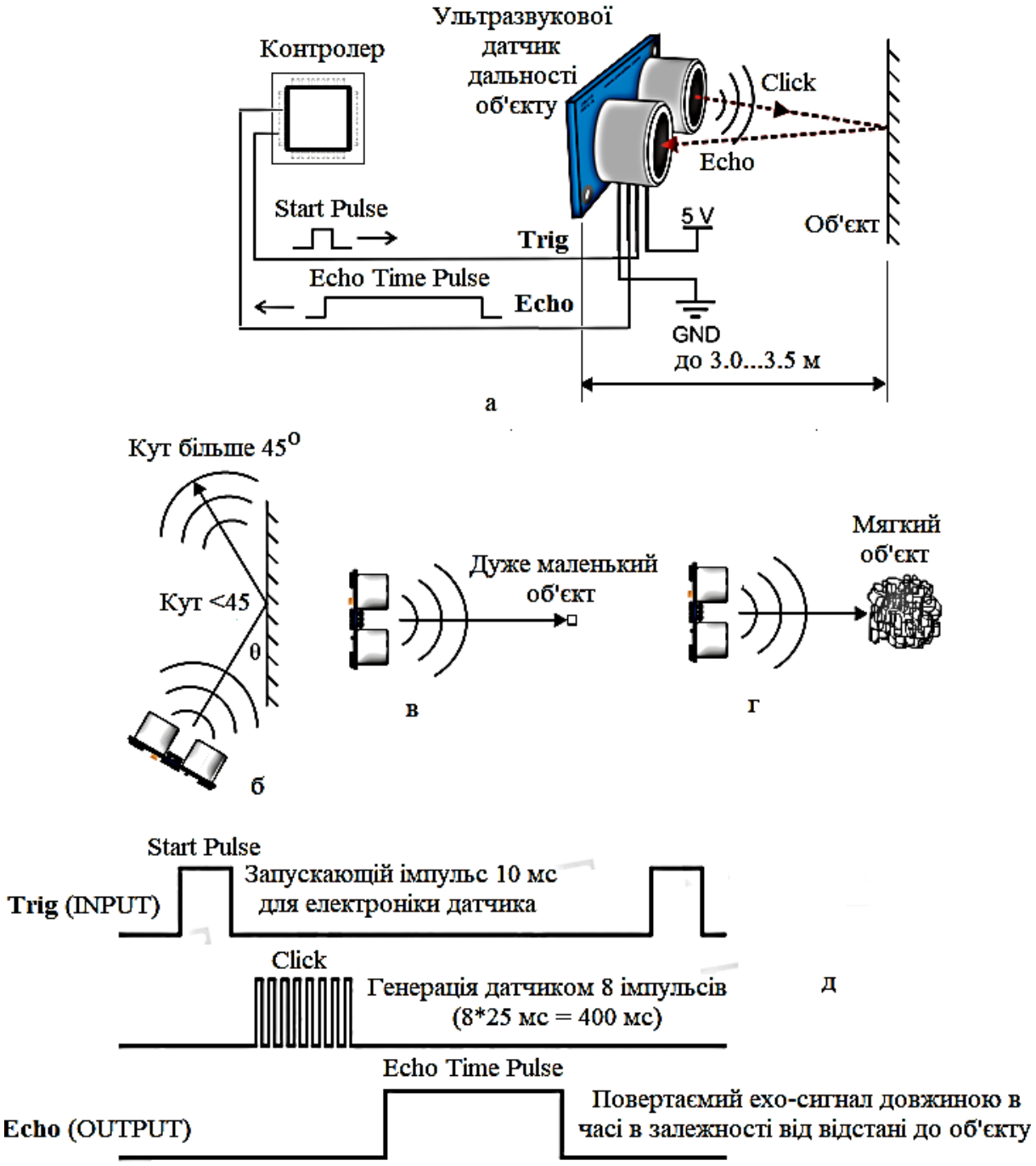


Рис. 7.24. Принципові схеми роботи і застосування ультразвукового датчика дальності: а – прямий і зворотний зв’язок датчика і об’єкту при допустимій відстані до 3,5 м; б – ехо-сигнал не повертається до датчика при куті більше 45 градусів до площині об’єкту; в і г – обмеження застосування датчика при малих розмірах об’єкту і при поглинанні ультразвукових хвиль пухким або м’яким об’єктом; д – сигнали на вході і виході датчика при його роботі

половини гри пінг-понг тому, що як і у в настільному тенісі відбувається click по кульці і очікування результату цього «кліку», тобто Echo-сигналу до наступного «кліку». Результат «кліку» розраховує гравець, а результат **click** по Echo-сигналу розраховує мікроконтролер контролера платформи **Arduino** або інших.

При розрахунках дальності до перешкоди, а саме відстані ультразвукового сигналу туди і назад потрібне переведення швидкості звуку в одиницях **м/с** в одиниці **сантиметр/мікросекунда**, що відбувається наступним чином. Щоб розрахувати відстань від Echo-сигналу, яка становить в два рази більше відстані до перешкоди (туди і назад), тому рівняння має наступний вигляд:

$$2S = c \cdot t \tag{7.5}$$

де с – швидкість звуку

$$\text{с} = 340 \text{ м/с} = \frac{340 \cdot 100}{10^6} = 0.0340 \frac{\text{см}}{\text{мкс}}.$$

В результаті отримаємо коефіцієнт 1/58 для програмного перерахунку довжини імпульсу Echo-сигналу в мікросекундах у відстань в сантиметрах за виразом (7.6):

$$S = \frac{c \cdot t}{2} = \frac{0.0340 \frac{\text{см}}{\text{мкс}} \cdot t}{2} = 0{,}017 \cdot t = \frac{t}{58} \tag{7.6}$$

для команди **`distance_sm = impulseTime/58;`**

Приклад підключення ультразвукового датчика HC-SR04 до контролера Arduino UNO наведений на рис,7.25. У датчика контакти **VCC, TRIG** (око T), **ECHO** (око R), **GND** з'єднані, відповідно, з пінами **+5V, D9, D8, GND** контролера.

При підключенні датчика за схемою на рис. 7.25 до контролера **Arduino,** закріпленого на робоплатформі і компіляції наведеного скетчу інформація о відстані буде відсилатися в порт комп'ютера, а також при відстані до датчика менше 30 см буде спрацьовувати світлодіод на платі, який задіяний до піну 13.

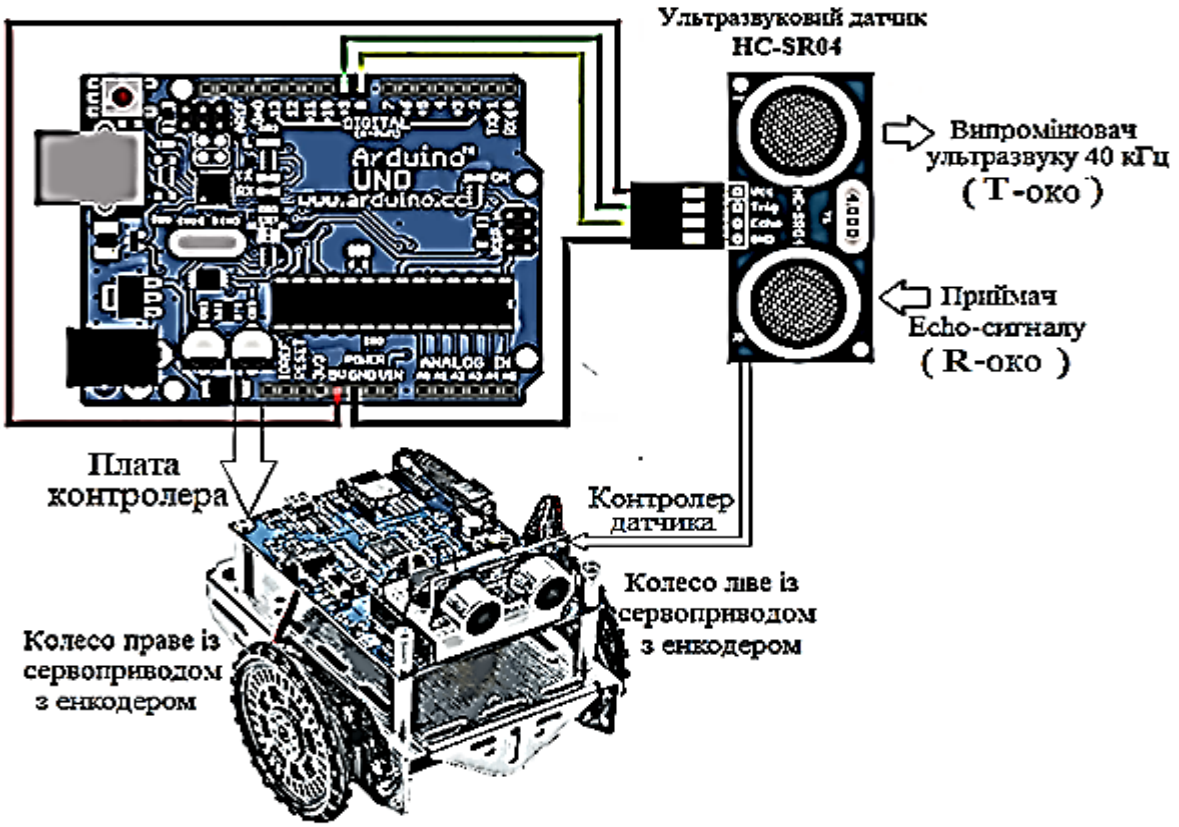


Рис. 7.25. Схема підключення ультразвукового датчика HC-SR04 до контролера Arduino UNO та три опорної рухомої робоплатформи з індивідуальним сервоприводами двох колес

Приклад програмного коду :

```
#define Trig 9
#define Echo 8
#define ledPin 13

void setup()
{
  pinMode(Trig, OUTPUT);          // Trig ініціюємо як ВИХІД
  pinMode(Echo, INPUT);           // Echo ініціюємо як ВХІД
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);             // задаємо швидкість 9600 бод/ с
                                  // спілкування з COM-портом
}
unsigned int impulseTime=0;    // непідписаний integer довжини
                               // імпульсу
```

```
unsigned int distance sm=0;   // непідписаний integer для
                              // виміру дальності об'єкту в см
void loop()
{
  digitalWrite(Trig,HIGH);     // включення входу Trig датчика
  delayMicroseconds(10);       // із затримкою на 10 мікросекунд
  digitalWrite(Trig,LOW);      // вимикання входу Trig датчика
  impulseTime=pulseIn(Echo,HIGH); // заміряємо довжину
                                  // імпульсу Echo-сигналу
  Distance sm=impulseTime/58;     // перераховуємо довжину
                                  // імпульсу в сантиметри
  Serial.println(distance sm); // відстань в см до перешкоди
//виводимо на екран робочого поля програми з підключеного COM-порту
  if (distancesm < 30)      // якщо відстань менше 30 сантиметрів
  {
    digitalWrite(ledPin, HIGH);// світлодіод піну 13 світиться
  }
  else                            // інакше
  {
    digitalWrite(ledPin, LOW);// світлодіод піну 13 не світиться
  }
  delay(100);                     //очікування 100 мс = 0.1 с

 /*наступний Echo-сигнал може бути створений, тільки після зникнення
попереднього Echo-сигналу. Це час називається періодом циклу (cycle
period). Рекомендований час очікування повинені бути не менше 50 мс між
стартовими імпульсами в 10 мкс для наступного Echo-сигналу */
}
```

Для постановки лабораторної роботи по розділу 7 розглянемо приклад підключення ультразвукового датчика, як далекоміра згідно з монтажною схемою (рис.7.26), побудованої в середовищі **Fritzing** і яка містить контролер Arduino UNO, макетну плату (Bread Board), датчик HC-SR04, два світлодіодf (ledRed і ledGreen), пьезодінамік (buzz) два резистора від 240 Ом і перемички («папа-папа»).

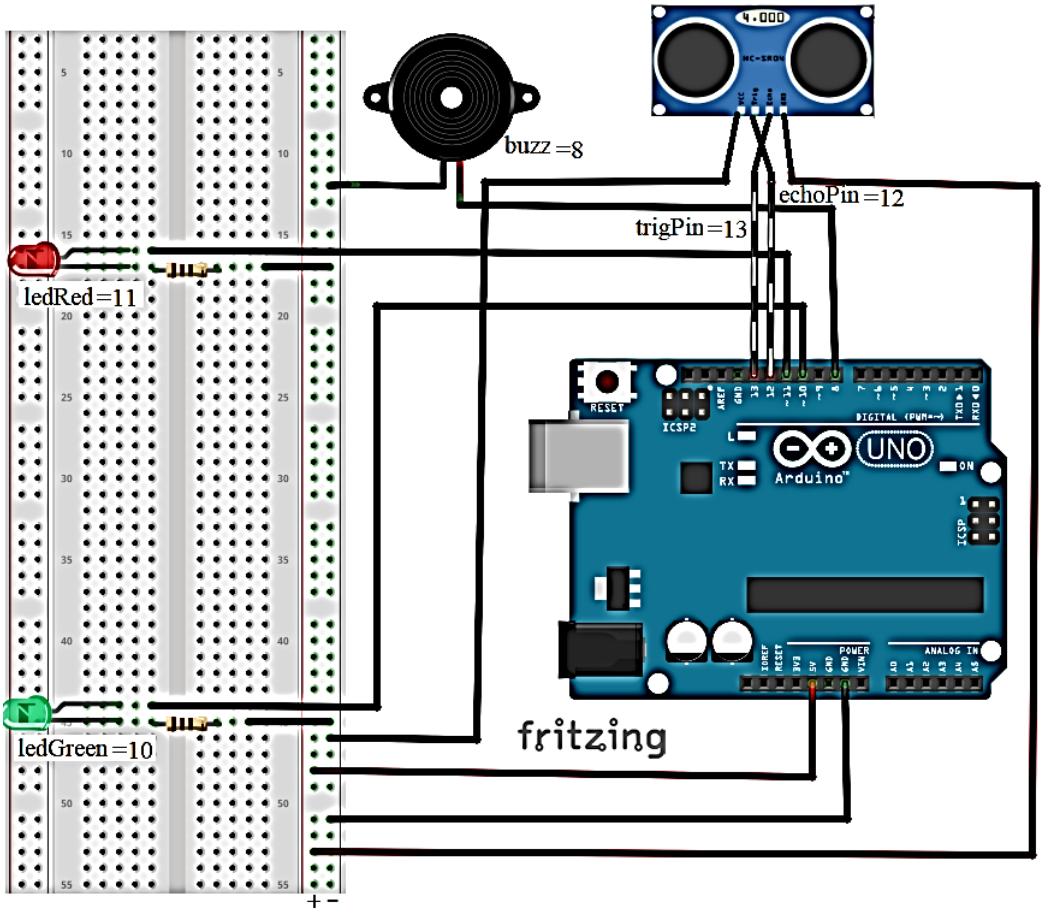


Рис. 7.26. Монтажна схема підключення ультразвукового датчика HC-SR04 до контролера Arduino UNO

Скетч для монтажної схеми на рис.7.26 має наступний вигляд:

/***цифрові піни і піни живлення контролера , до яких підключені компоненти схеми на рис.7.26***/

```
int trigPin = 13;
int echoPin = 12;
int ledRed = 11;
int ledGreen = 10;
int buzz = 8;
```

/*** команди між двома фігурними дужками **{...}** – функція **void setup()** це процедура установки секції коду (**pinMode** - режим піна), що діє одноразово при запуску програми ***/

```
void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
    pinMode(ledRed, OUTPUT);
    pinMode(ledGreen, OUTPUT);
    pinMode(buzz, OUTPUT);
    }
```

/*** команди між двома фігурними дужками {...} – функція **void loop()** це команди циклу для процедури перевірки пінів ВХОДІВ і пінів ВИХОДІВ та процедури розрахунку відстані до перешкоди за тривалістю ехо-сигналу з урахуванням швидкості звуку. Цикл виконується постійно поки працює контролер ***/

```
  void loop() {
    long duration, distance;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) / 29.1;
    if (distance < 20)
    {
    digitalWrite(ledRed,HIGH);
    digitalWrite(ledGreen,LOW);
    tone(buzz, 1000);
    }
```

/* команда **else** «інакше» для секції коду і команда **if** «якщо» для контролю відстані від датчика до перепони */

```
    else
    {
    digitalWrite(ledRed,LOW);
    digitalWrite(ledGreen,HIGH);
    noTone(buzz);
    }
    if (distance >= 200 || distance <= 0)
    {
    Serial.println("Out of range");
    }
    else {
```

```
Serial.print(distance);
Serial.println(" cm");
}
delay (500);
}
```

7.9. **Програмування аналогового датчика освітленості на платформі Arduino**

У контролері Arduino UNO немає пінів аналогових виходів, а є тільки піни входів (A0…A5) аналогових сигналів з аналогових датчиків (температури , вологості , концентрації рідини , терморезисторів, тензорезисторів та інших). Контролер має вбудований аналого-цифровий перетворювач (АЦП) і не має цифро-аналогового перетворювача (ЦАП). Але деякі цифрові піни 0…13, які мають позначку тільда «~» можуть працювати, як аналогові виходи. Для цього після підключення до пінів ~ 3, ~5 , ~6 , ~9 , ~10 , ~11 в програмі треба передбачити наступну відповідність.

Аналогові входи мають АЦП 10-розрядне (2^{10}) = 1024, а 8-розрядні цифрові (аналогові) виходи можуть приймати значення от 0 до $(2^8 - 1)$ = 255, тобто до числа 256 значень. Тому значення на аналогових входах потрібно визначати як 1024/256 = 4 і команда при використанні, наприклад, піна ~ 3 для квазіаналогового сигналу має вигляд: «**analogWrite (3, 256-(a/4));»,** а =1024).

Електрична схема дільника напруги (рис. 7.27,а), схема з'єднань елементів на макетної платі (рис. 7.27,б) і програма зміни освітленості фоторезистора, який під'єднаний до аналогового входу A1 плати Arduino UNO (рис. 7.27,в) наведені на рис. 7.27.

Функція **Serial.begin(9600);** відкриває COM-порт на екране після активізації кнопки «Монітор порту» у правої стороні стрічки меню (рис. 7.28,а), у який заходиться 5 кнопок з лівої сторони цієї же стрічки меню. При цьому встановлюється швидкість для послідовної передачі даних за допомогою відповідного (9600 бод) апаратного інтерфейсу UART і на екран виводиться колонки значень чисел, які відповідають значенням перетвореного з аналогового неелектричного сигналу з датчика у відповідний цифровий електричний сигнал. Якщо 1 кадр передачі сигналу складається з 8 біт, плюс 1 біт стартовий, плюс 1 біт парності, плюс 2 біта

стоп-кадра, тоді швидкість для послідовної передачі сигналу для 9600 бод в біт/с дорівнюється (8+1+1+2)*9600=115200 біт/с.

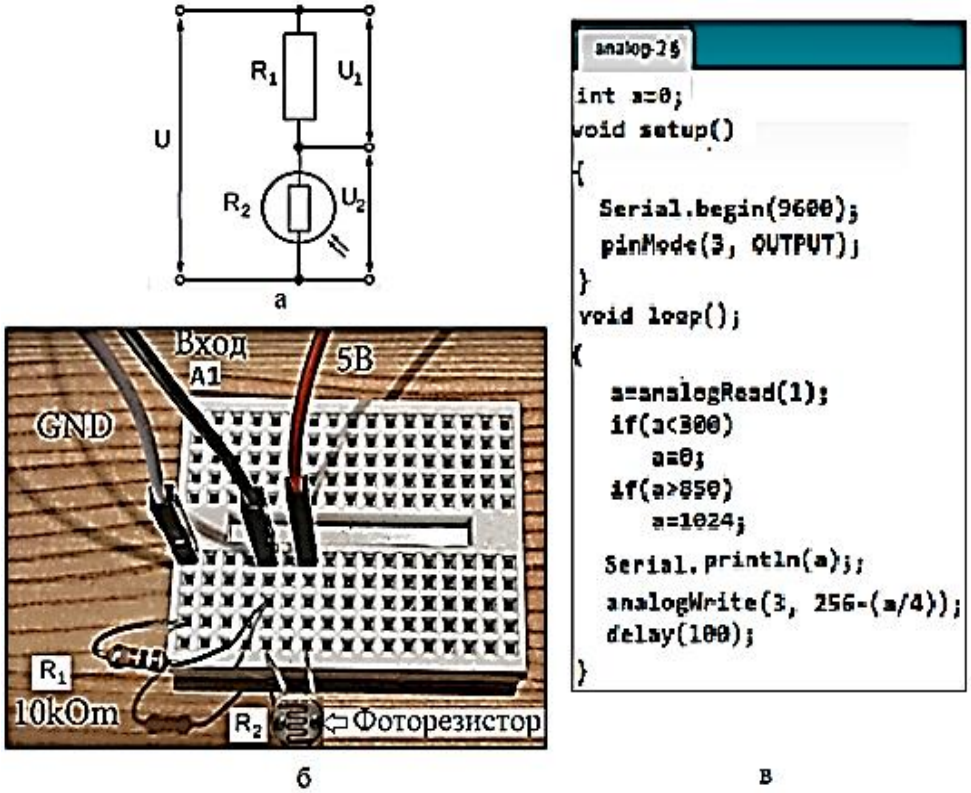


Рис. 7.27. Схеми підключення аналогового датчика (фоторезистора) – рис. а, рис. б і програма зміни освітленості фоторезистора (рис. в)

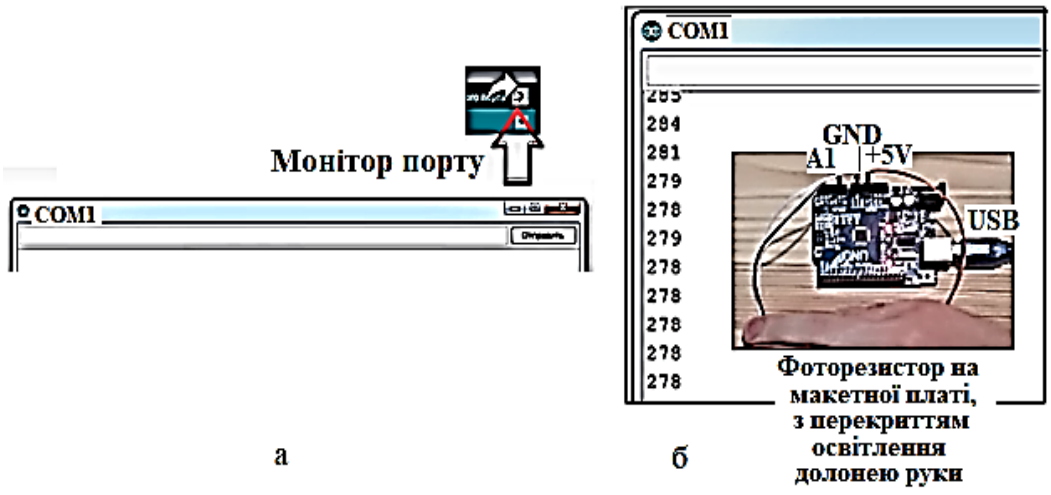


Рис. 7.28. Відкриття монітору послідовного порту COM1 (рис. а) і результати зміни освітленості фоторезистора, які виводяться на монітор порту COM1(рис. б)

Питання для самоконтролю по розділу 7

1. Як розшифрувати і яке призначення ADC і DAC в схемах мехатроніки і робототехніки ?

2. Який принцип дії і яке призначення енкодера і датчика Холла в схемах мехатроніки ?

3. Що таке код Грея і як він пов'язаний з кутом повороту головного валу технологічних машин ?

4. Як програмуються таймери в CoDeSys?

5. Як програмуються лічильники в CoDeSys ?

6. Що означає команда **`NOP`** та коли і що відбувається в програмі при дії команди **`NOP`** ?

7. Чим відрізняється таймер **TOF** від таймера **TON** ?

8. Що означає **CPU** і **CTD** і чим вони відрізняються ?

9. Як працює ультразвуковий датчик і галузі його застосування в схемах мехатроніки ?

10. Що означає команда **`digitalWrite(Trig,HIGH`** і які наведені помилки в запису цієї команди ?

11. Яке призначення фігурних і круглих дужок в скетчах (sketches) ?

12. Яке призначення наступних позначень **//, /* */, ;** та **#** в скетчах ?

8. БІСТАБІЛЬНЕ І МОНОСТАБІЛЬНЕ КЕРУВАННЯ ЦИКЛАМИ БЕЗ ПРОГРАМУЄМОГО ЛОГІЧНОГО КОНТРОЛЕРА

8.1. **Побудова комбінованих схем мехатроніки у програмному середовищі FluidSim-P і в програмному середовищі Fritzing**

Для реалізації проектів мехатроніки без програмуємого логічного контролера використовуються електромагнітні, електронні і пневматичні елементи і модулі. Для імітації роботи системи мехатроніки в режимі реального часу і аналізу роботи циклу доцільно використовувати програмне середовище **FluidSim-P**. При роботі з гідравлічними елементами мехатроніки при створенні проектів використовують програмне середовище FluidSim-**H** (FluidSim-**H**ydraulics).

Порядок роботи в програмному середовищі FluidSim-P:

Після встановлення програми і відкриття файлу New Project на моніторі відображається інструментальна панель елементів і робоче поле для створення нового проекту, як зображено на рис. 8.1.

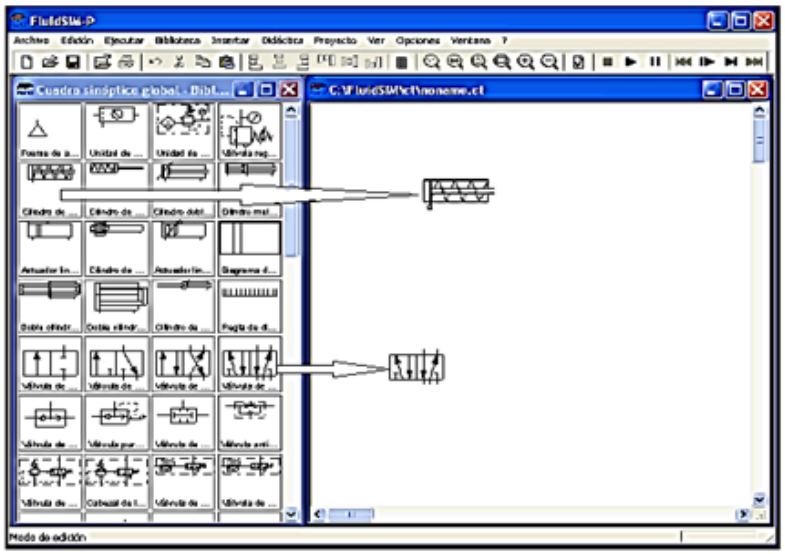
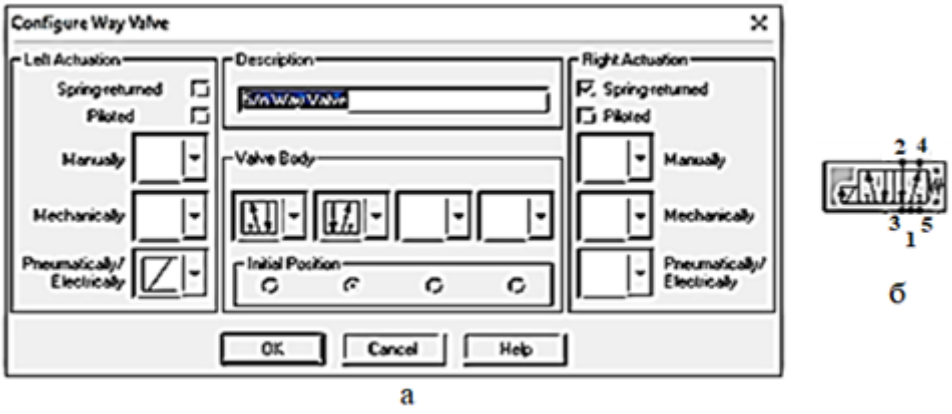


Рис. 8.1. Головне меню і приклад перетягування з інструментальної панелі елементів на робоче поле проекту в програмі **FluidSim-P**



а б

Рис. 8.2. Приклад меню діалогу (а) додавання конструктивних особливостей обраного пневмо-розподільника і результат (б) двох реалізованих опцій

Fritzing – це програмний інструментом розробника з відкритим вихідним кодом для навчання, прототипування і обміном проектами на базі Arduino. Програма полегшує процес прототипування проектів на базі популярних платформ: **Arduino**, **Raspberry Pi** та інших. Програма містить велику кількість віртуальних моделей різних платформ контролерів, компонентів і модулів, які можна розставляти на робочому полі і підключати до макетної плати, створюючи таким чином монтажну електричну схему, електричну принципову схему і друковану схеми до мехатронного пристрою. Програма має з великою кількістю бібліотек різних елементів мехатроніки, деякі умовні позначення яких наведені в таблиці 1.1.

Після завантаження програми **Fritzing** з’являється вкладки і панель інструментів (рис.8.3).

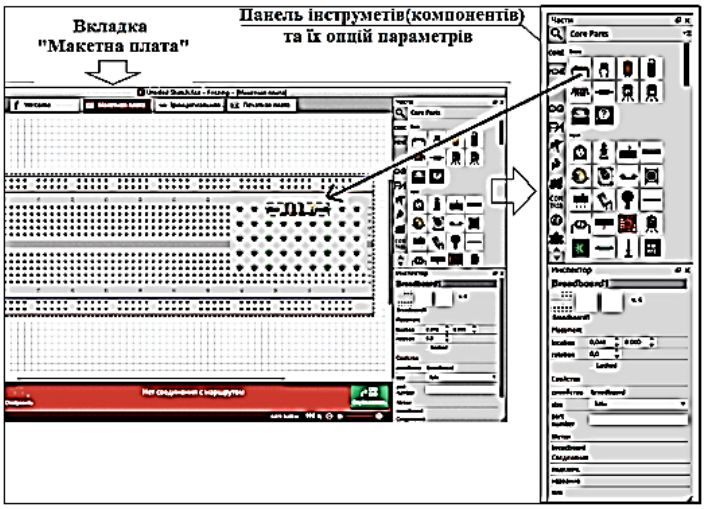


Рис.8.3. Фрагмент схеми перетягування елементів мехатроніки з інструментальної панелі на макетну плату в програмі **Fritzing**

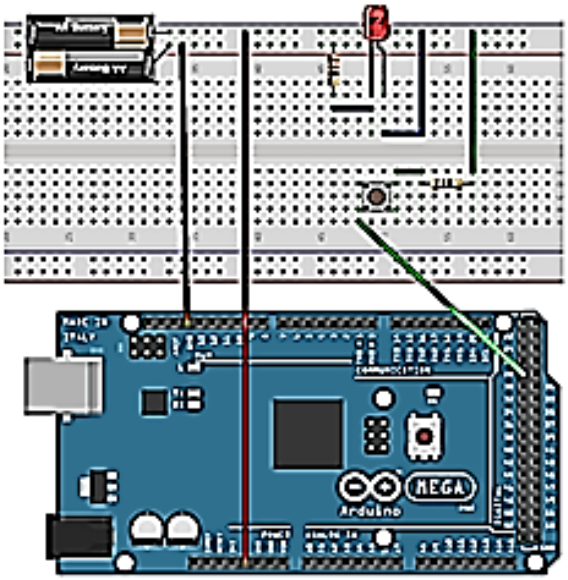


Рис.8.4. Приклад монтажної електричної схеми в середовищі **Fritzing**

Потрібні елементи перетягуються з панелі інструментів на макетну плату так, як це виконувалося з елементами для пневматичної і електричної схем при роботі в програмному середовищі Fluid Sim. Виводи елемента повинні співпадати з отвором плати. При цьому курсору миші наведений в цю точку буде синього кольору, а стовпчик контактів виводів перетягнутого елементі будуть мати зелений колір. Для з'єднання отворів на макетної платі дротами (перемичками) утримуємо ліву клавішу миші і протягуємо курсор миші на наступний отвір макетної плати (контакт схеми). І так далі перетягуємо наступні елементи схеми, які з’єднаємо віртуальними дротами (провідниками). В результаті отримуємо монтажну електричну схему ,наприклад, що наведена на рис.8.4.

8.2. **БІстабільне керування виконавчим механізмом з двома кінцевими вимикачами для виконання циклу «1–N1»**

Розглянемо складання найпростіших команди для роботи системи, що складається з пневматичного циліндру 1, пневморозподільника 4 типу 5/2 з електромагнітним БІстабільним керуванням, дроселів 2 і 3, двох кінцевих вимикачів XN1 і X1 і та кнопкою «Start» S1 (рис. 8.5).

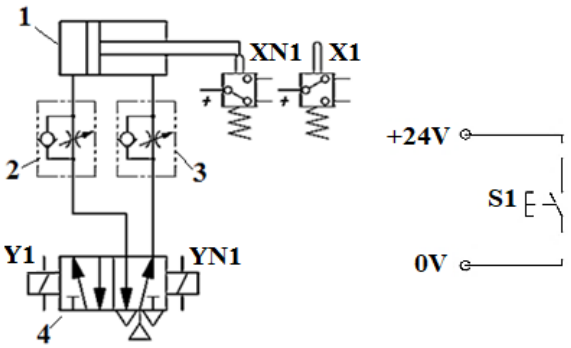


Рис. 8.5. Схема типового мехатронного модуля з БІстабільним керуванням

Кінцевий вимикач (кінцевик) **XN1** відповідає за втягнуте положення штока поршня, а кінцевик **X1** відповідає за витягнуте положення. При натисненні штока на кінцевик замикається електричний контакт. У витягнутому положенні штока логічні змінні **X1=1** (контакт кінцевика замкнений) і **XN1=0** (контакт кінцевика розімкнений). Електромагніт золотника (клапана), що відповідає за прямий рух поршню пневмоциліндру позначимо **Y1**, а електромагніт, що відповідає за зворотній рух позначимо **YN1**. При подачі сигналу на електромагніт **Y1** його логічна змінна **Y1=1**, при знятті сигналу **Y1=0**. При подачі сигналу на електромагніт **YN1** його логічна змінна **YN1=1**, при знятті сигналу **YN1=0**.

Нехай необхідно реалізувати цикл в якому циліндр буде виконувати виштовхування штока до кінцевика **X1**, а потім назад до кінцевика **XN1.**

Домовимось, що такий цикл позначається наступним чином «**1–N1**». Після натискання кнопки «Start» **S1** сигнал буде поданий на котушку електромагніту **Y1**. Тоді пряма команда на спрацювання електромагніту **Y1** матиме вигляд:

$$\mathbf{Y1 \Leftarrow XN1} \qquad (8.1)$$

Після того, як шток пневмоциліндру наїде на кінцевик **X1**, необхідно подати сигнал на електромагніт **YN1** для того, щоб поршень зі штоком почав зворотній рух, тобто зворотна командою для пневмоциліндру буде:

$$\mathbf{YN1 \Leftarrow X1} \qquad (8.2)$$

При створенні будь-якої циклової системи керування необхідно передбачити сигнал для початку роботи системи. Тому в першу команду

додаймо умову, що для початку роботи системи повинна бути натиснутою кнопка **S1**. Отже команди для роботи системи по циклу «**1–N1**» мають вигляд:

$$\mathbf{Y1 \Leftarrow XN1 \cdot S1}; \qquad (8.3)$$
$$\mathbf{YN1 \Leftarrow X1}.$$

Зауваження. При керуванні БІстабільним пневморозподільником недопустимим щоб одночасно виконувались умови **Y1=1** та **YN1=1**.

Складання електричної схеми керування за логічними командами.

Для початку необхідно представити на схемі всі сигнали, які присутні в мехатронній системі. В нашому випадку це сигнали S1, XN1 та X1, кожен з яких необхідно завести на окреме електричне реле, оскільки сигнал може використовуватись багаторазово (рис. 8.6).

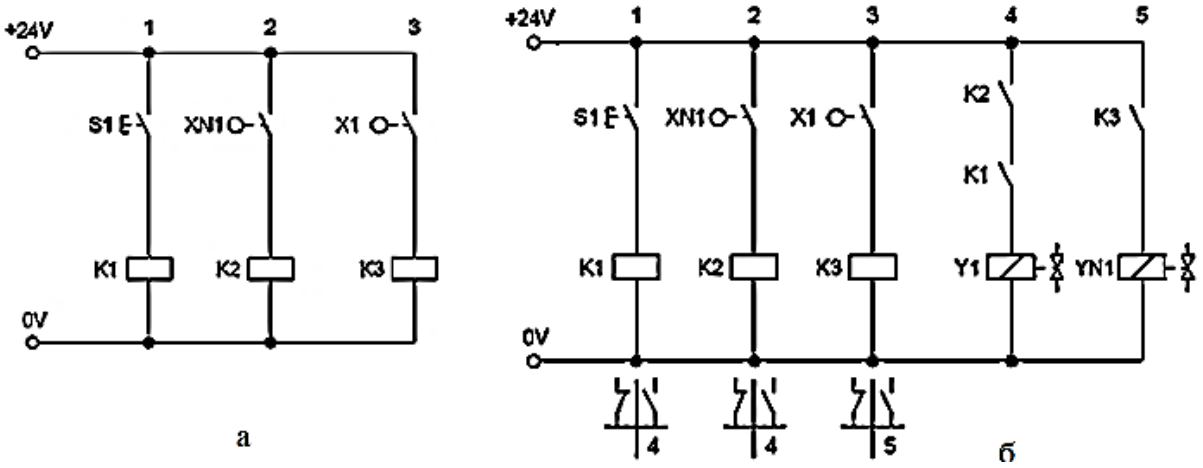


Рис. 8.6. Фрагменти електричної схеми (тип Е3) для вхідних сигналів - кнопки пуск S1 і кінцевих вимикачів XN1 та XN1: а – прямого включення електромагнітів Y1 і YN1; б – непрямого включення електромагнітів Y1 і YN1

Далі додаємо до ланцюгів 1, 2 і 3 на схеми на рис.8.6,а електричні ланцюги 4 і 5 на рис. 8.6,б у які входять контакти вхідних сигналів, що відповідають прямій команді для електромагніту Y1 та зворотній команді для електромагніту YN1. Після натискання кнопки S1 сигнали керування будуть почергово подаватися на відповідні котушки електромагнітів Y1 та YN1 пневморозподільників в ланцюгах 4 і 5.

Виконуємо імітацію роботи схеми на рис. 8.6,б з використанням двох кнопок S1 і S2 без фіксації (рис. 8.7) у натиснутому стані. Комбінована

схема на рис.8.7 непрямого включення електромагнітів Y1 і YN1 пневморозподільника працює наступним чином. Натискаємо кнопку S1 без фіксації у натиснутому стані.. Спрацьовує реле К1 і кнопка S1 через перший контакт К1, який замикається береться на самоблокування в ланцюгу 2. Через другій контакт К1 утворюється замкнений ланцюг 5 для проходження електричного струму через котушку електромагніту Y1 (реалізована вручну пряма команда) і шток з поршнем пневмоциліндру висуваються «вперед». Після відключення реле К1 і натисканні кнопки S2 дії повторюються, але вже з реле К2 і електромагнітом YN1. Результатом є повернення штоку з поршнем пневмоциліндру «назад» (реалізована вручну зворотна команда). При включенні електромагнітів Y1 і YN1 БІстабільного пневморозподільника світиться його індикатор і чути одноразове клацання золотника відповідного електромагніту.

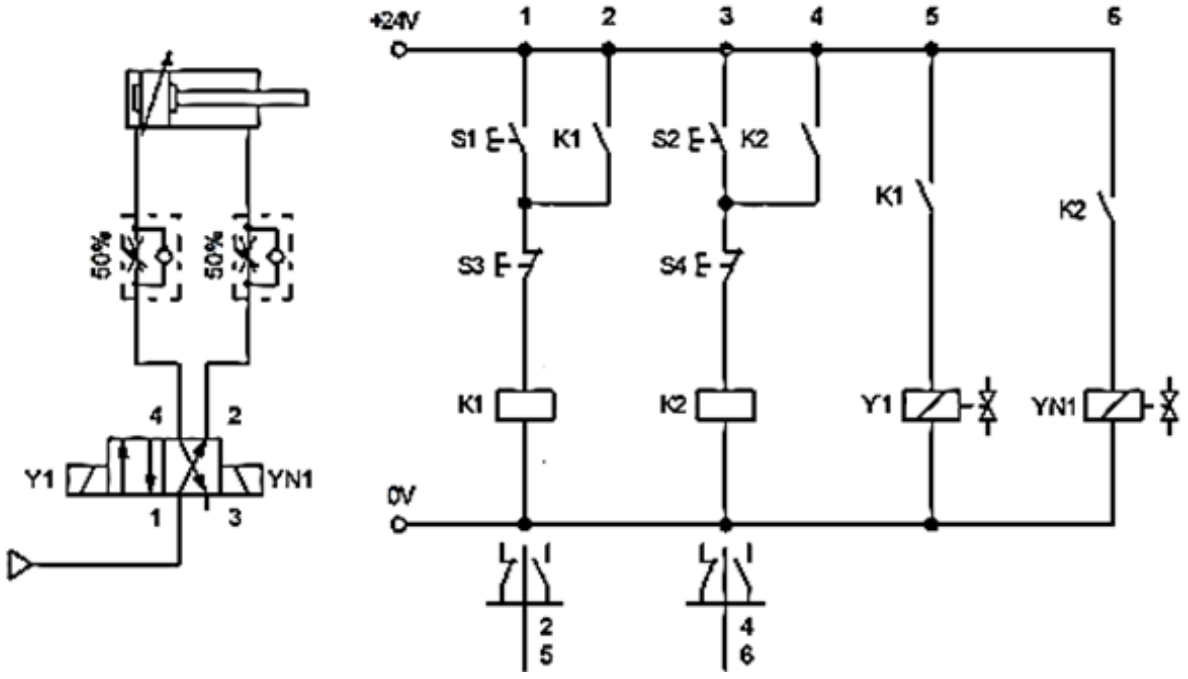


Рис. 8.7. Комбінована схема керування БІстабільним пневморозподільником 4/2 з електромагнітами Y1 і YN1

На рис. 8.8 наведена комбінована схема (тип С3=П3+Е3) для реалізації циклу «**1–N1**» з БІстабільним керуванням та двома кінцевими вимикачами, яка відтворює роботу типового кривошипно-повзунного механізму чи кривошипно-коромислового механізму без кривошипу і без шатуна. При цьому, якщо на штоку пневмоциліндру двосторонній дії буде

закріплена голка – отримаємо аналог механізму голки швейних машин, а якщо на штоку пневмоциліндру закріпити пластинчастий ніж – отримаємо аналог пересувної розкрійної машини. Після виконання схеми в програмному середовищі **FluidSim** виконуємо анімацію її роботи при натисканні на кнопку Start S1. Далі відбувається наступна взаємодія і робота пневматичних і електричних елементів схеми згідно з наступними виразами:

1. Пряма команда на висування штоку (в дужках позначені ланцюги електричної схеми):

S1→ S2→ (реле К1) + К1(2) + К1(3) + К1(4) → (реле К2) + К2(5) → Y1:= "1" (8.4)

2. Зворотна команда на втягування штоку:

X1 →К1(4) →(реле К3) + К3(6) → YN1:= "1" (8.5)

3. Повторювання циклу «**1–N1**» відбувається по п.1 і по п.2 до п.4.

4. Зупинення циклу: **S3** → схема переводиться в стан яке на рис. 8.8.

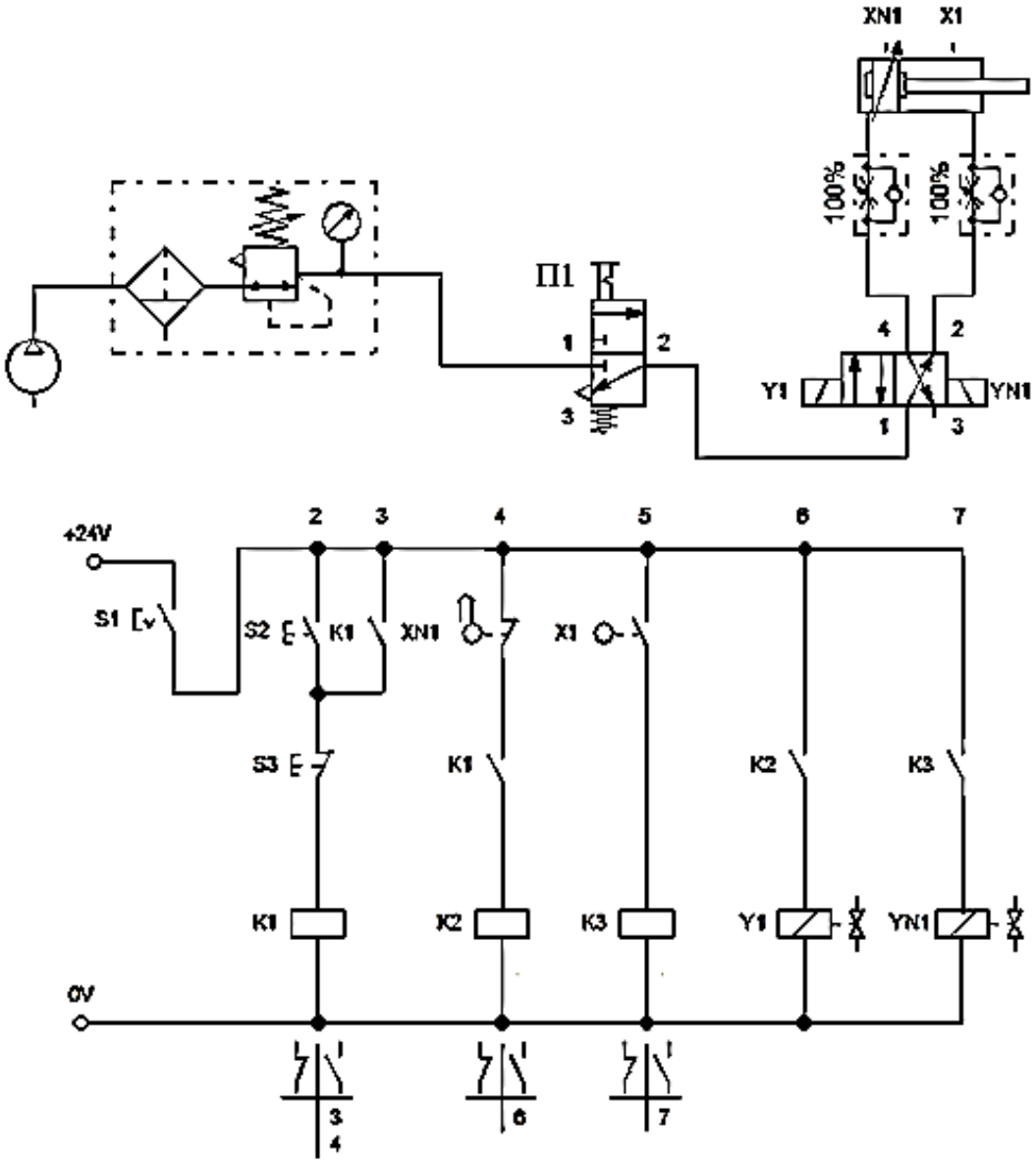


Рис. 8.8. Комбінована схема (тип С3=П3+Е3) циклу «**1–N1**» з БІстабільним керуванням та двома кінцевими вимикачами **XN1** та **X1**

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.8. треба відповісти на наступні питання.

Яка помилка і чим відрізняються крім помилки схема наведена на рис. 8.9 від схеми на рис. 8.8 ?

Для якого такту циклу на рис. 8.10 наведено скриншот анімації циклу роботи схеми ?

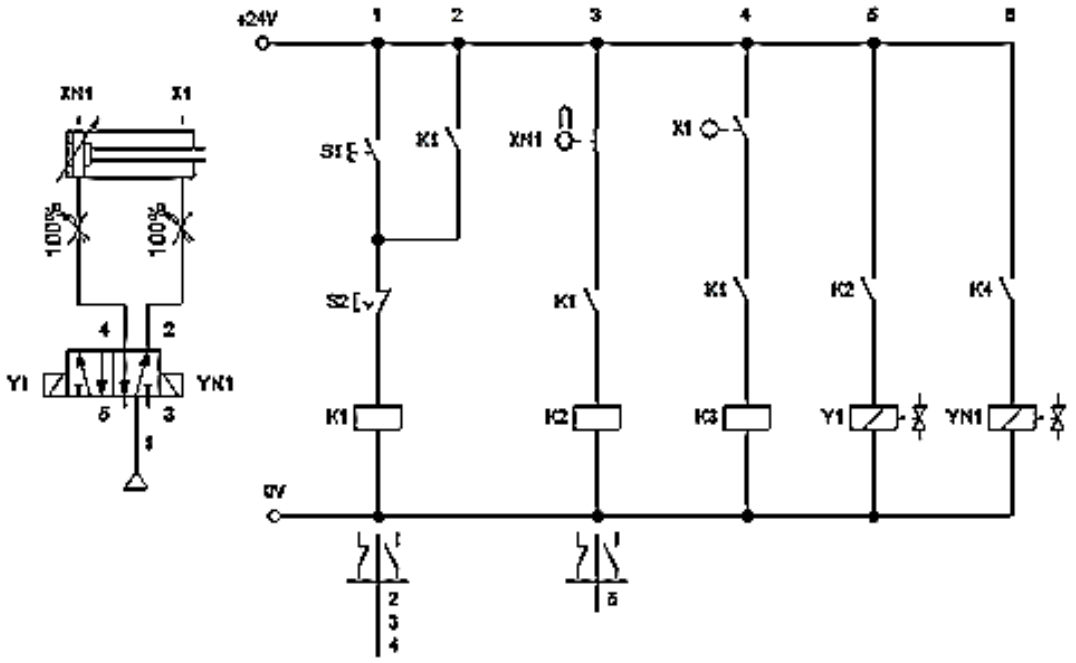


Рис. 8.9. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням та двома кінцевими вимикачами (з помилкою)

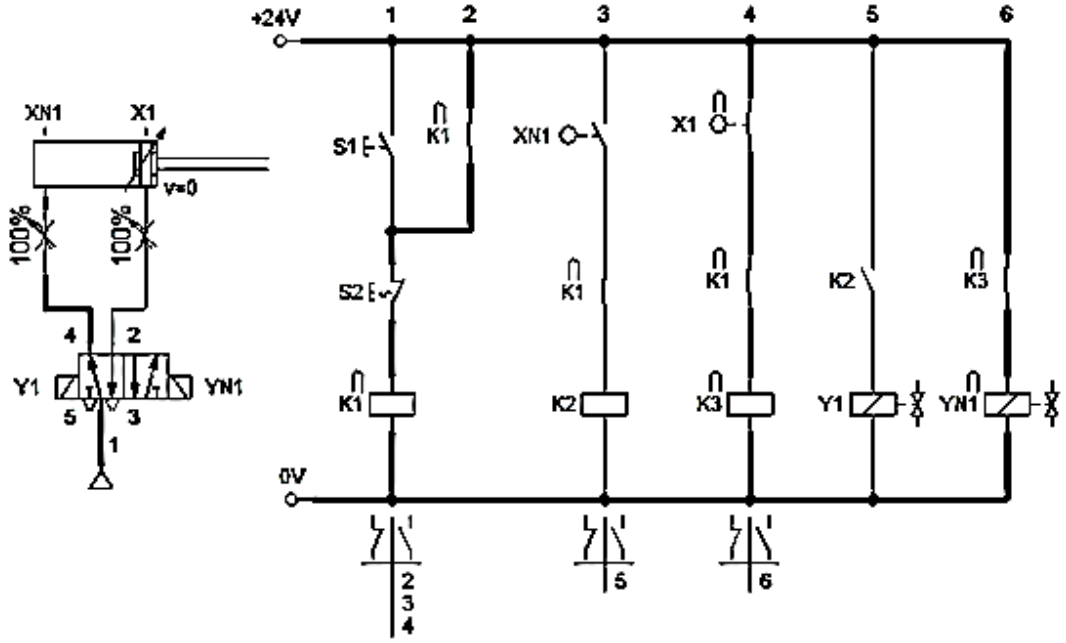


Рис. 8.10. Скриншот анімації циклу «**1–N1**» з БІстабільним керуванням та двома кінцевими вимикачами

8.3. **МОНОстабільне керування виконавчим механізмом з двома кінцевими вимикачами для виконання циклу «1–N1»**

На рис.8.11 і рис.8.12 наведені два варіанта комбінованої схеми для реалізації циклу «**1–N1**» в програмному середовищі *FluidSim* для реалізації циклу «**1–N1**» з МОНОстабільним керуванням та двома кінцевими вимикачами. Варіант 1 - з фіксацією кнопки S1 і і п'ятьма проміжними реле **К1…К5**, а варіант 2- з кнопкою S1 із блокуванням контактом реле К1 та трьома проміжними реле **К1…К3**.

Виконання циклу роботи (*для схеми* за *варіантом 1*) виконується згідно з наступними виразами:

1. Алгоритм виконання прямої команди на висування штоку:

$$\mathbf{S1 \rightarrow (\text{ реле К1}) + К1(2) + К1(4) + К1(6) \rightarrow (\text{ реле К2}) + К2(4) - К2(6) \rightarrow}$$

$$\mathbf{\rightarrow (\text{реле К4}) + К4(5) + К4(8) \rightarrow Y1 := "1"}, \qquad (8.6)$$

де Y1:= "1" - електромагніт пневморозподільника включений;

знак «плюс» означає замикання контакту реле і включення котушки відповідного електромагнітного реле чи електромагніту пневморозподільника до джерела струму;

знак «мінус» означає розімкнення контакту реле і відключення котушки відповідного електромагнітного реле чи електромагніту пневморозподільника від джерела струму.

2. Зворотна команда при МОНОстабільному керуванні відсутня. Повернення поршню і штоку назад відбувається при розмиканні контакту К4(8) і як наслідок переривання струму у котушці електромагніту Y1 пневморозподільника. При цьому під дією пружини відбувається переміщення золотника пневморозподільника у стан, який наведений на рис. 8.12. Це пояснює наступний вираз:

$$\mathbf{X1 := "1" \rightarrow XN1 := "0" \rightarrow (\text{реле К3}) + К3(6) \rightarrow (\text{реле R5}) - К5 \rightarrow}$$

$$\mathbf{\rightarrow (-\text{релеК4}) - -К4(8) \rightarrow YN1 := "0"}, \qquad (8.7)$$

де YN1:= "0" - електромагніт пневморозподільника вимкнений.

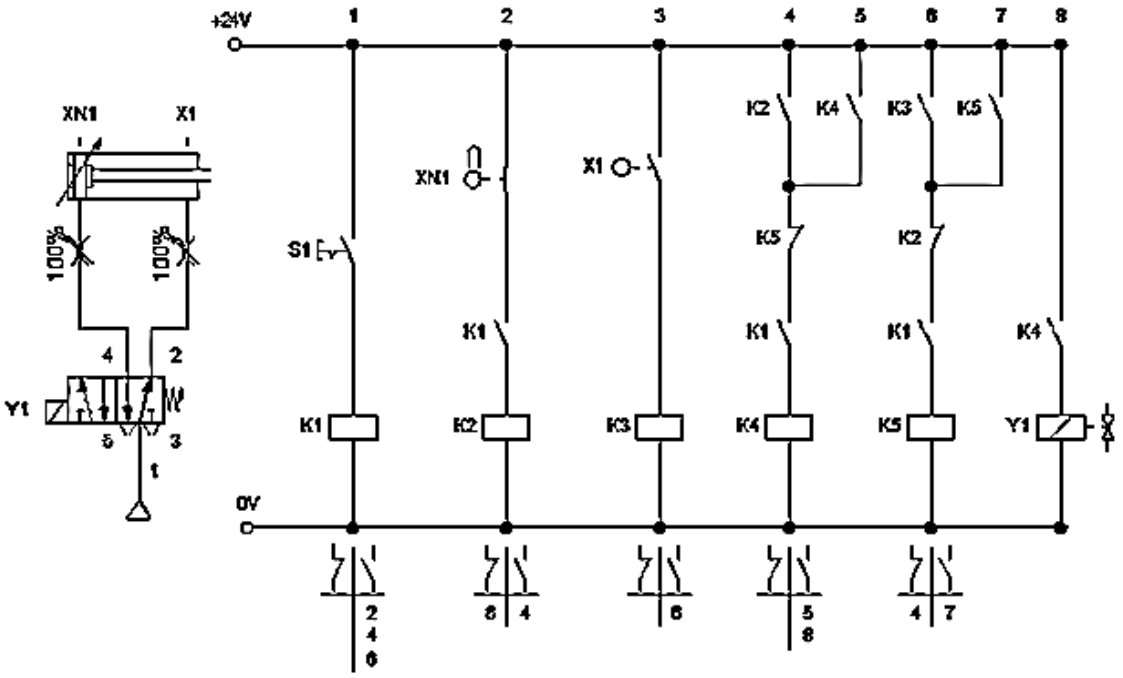


Рис. 8.11. Комбінована схема циклу «**1–N1** з МОНОстабільним керуванням та двома кінцевими вимикачами **XN1** і **X1** (*варіант 1*)

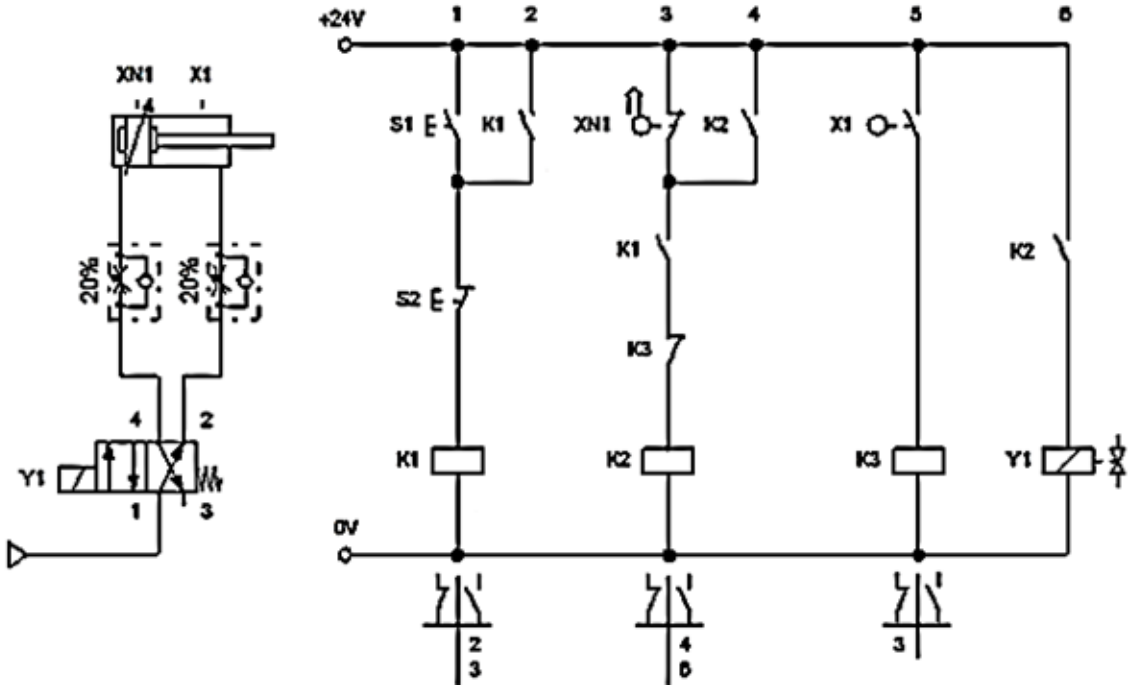


Рис. 8.12. Комбіновані схеми циклу «**1–N1**» з МОНОстабільним керуванням та двома кінцевими вимикачами **XN1** і **X1** (*варіант 2*)

3.Повторювання циклу **1–N1** відбувається по **п.1** і по **п.2** до **п.4.**

4.Зупинення циклу: повторне натискання кнопки **S1.**

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.12. треба відповісти на наступні питання.

Чим відрізняється схема на рис. 8.13 від схеми на рис. 8.11 ?

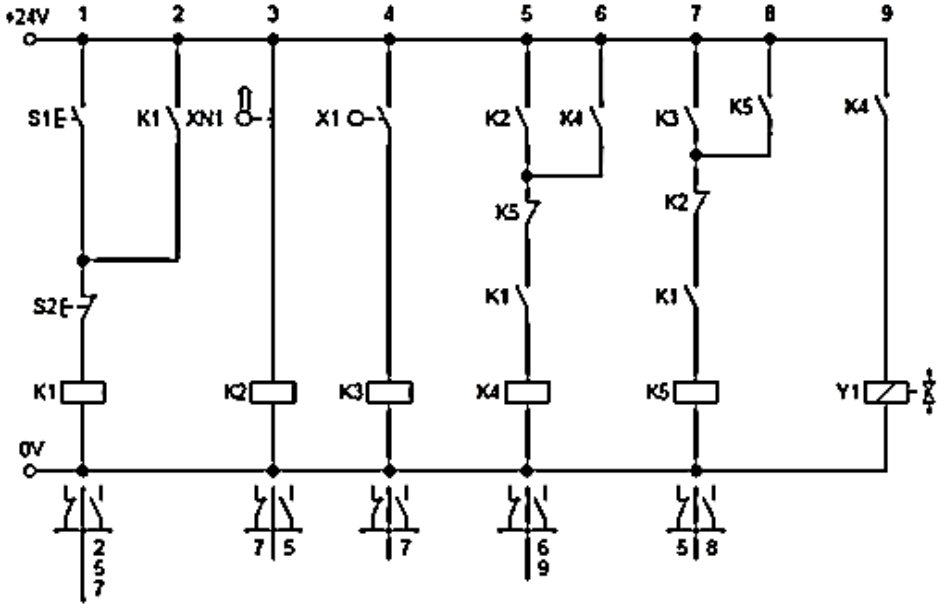


Рис. 8.13. Комбінована схема циклу «**1–N1** з МОНОстабільним керуванням та двома кінцевими вимикачами **XN1** і **X1**

Які помилки наведені в схемі на рис.8.14 ?

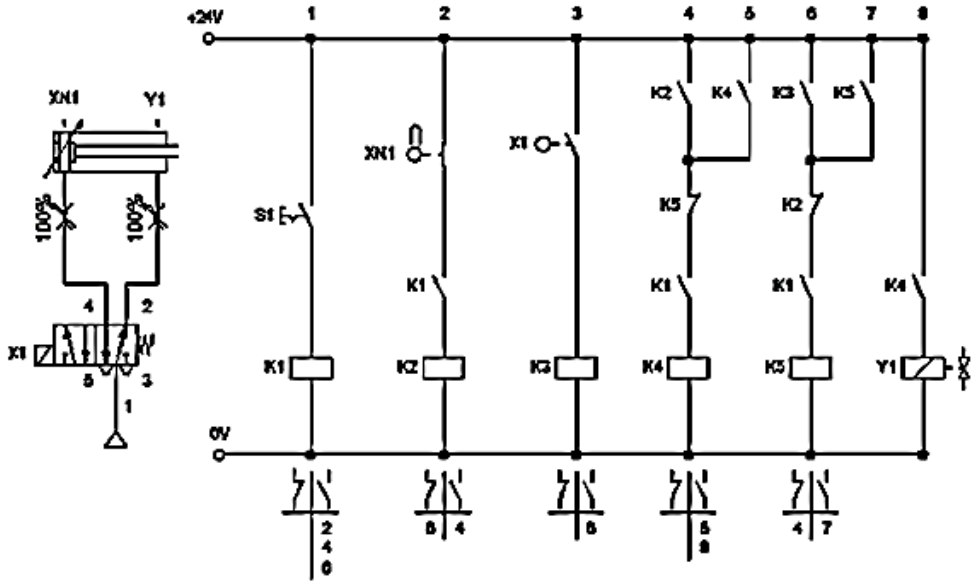


Рис. 8.14. Комбінована схема циклу «**1–N1** з МОНОстабільним керуванням та двома кінцевими вимикачами **XN1** і **X1**

Для якого такту циклу наведений скриншот анімації циклу роботи схеми на рис. 8.15 ?

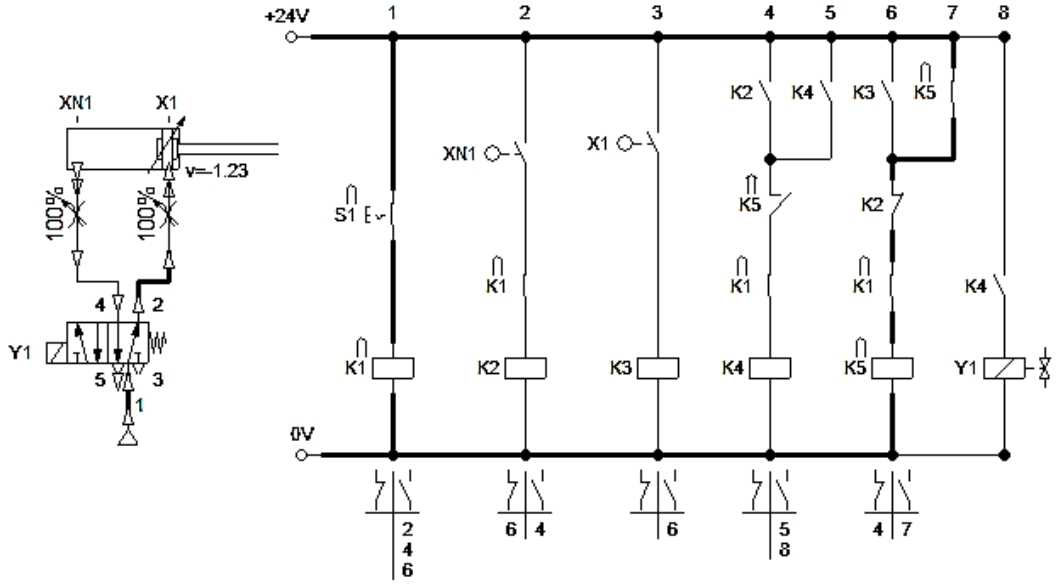


Рис. 8.15. Скриншот анімації циклу «**1–N1**» з МОНОстабільним керуванням та двома кінцевими вимикачами

8.4. **БІстабільне керування виконавчим механізмом з одним кінцевим вимикачем для виконання циклу «1–N1»**

На рис.8.16 наведена комбінована схема для реалізації циклу $1 \to \bar{1}$ в програмному середовищі FluidSim з БІстабільним керуванням та з одним кінцевим вимикачем. і виконана анімація її роботи (рис. 8.19).

1.Алгоритм виконання прямої команди на висування штоку:

S1 →(реле К1) +К1(3) +К1(4) +К1(7) →(реле К2) +К2(9) →Y1:= "1"
YN1:= "0" (8.8)

2.Алгоритм виконання зворотної команди на втягування штоку:

X1 → реле К3) + К3(6) + К3(7) + К3(10) - К3(4) → (реле часу К4)+К4(8)_t=0.2 с) → (реле К5) -К4 -К6 →YN1:= "1"
Y1:= "0" (8.9)

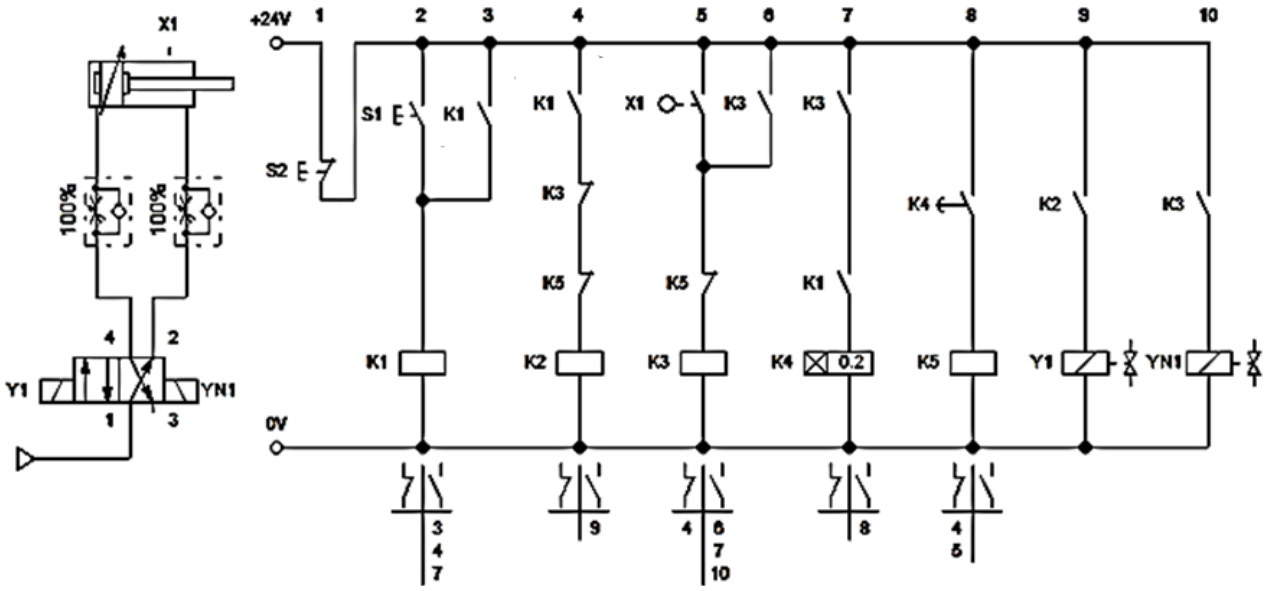


Рис. 8.16. Комбінована схема циклу $1 \rightarrow \bar{1}$ з БІстабільним керуванням та з одним кінцевим вимикачем

3. Повторювання циклу 1–N1 відбувається по **п.1** і по **п.2** до **п.4**

4. Натискання кнопки **S2**→ зупинення циклу → схема переводиться в стан як зображено на рис. 8.16.

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.16. треба відповісти на наступні питання.

1.Чим відрізняється неробоча схеми на рис. 8.17 від робочої схеми на рис. 8.16?

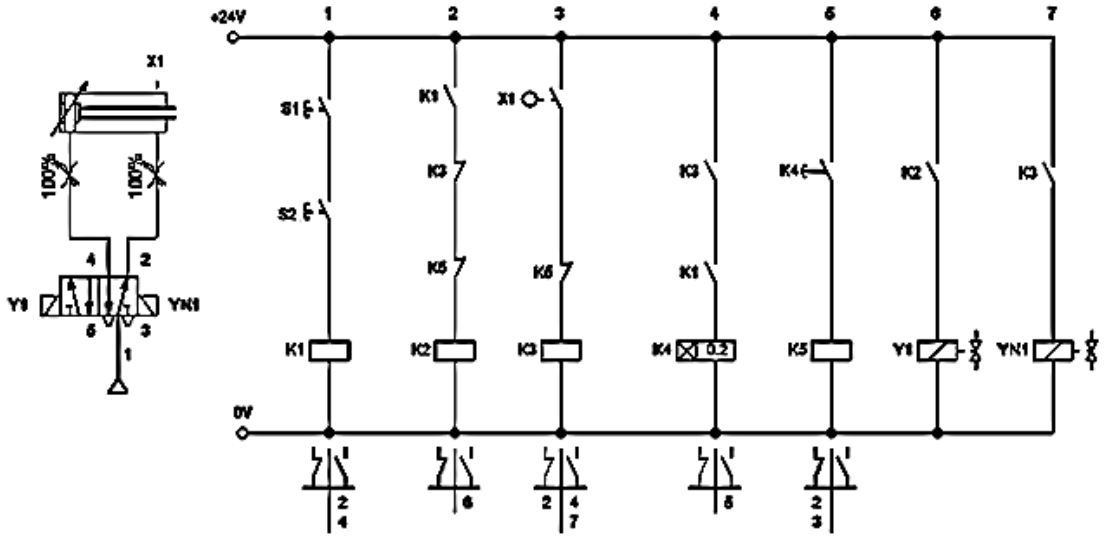


Рис. 8.17. Комбінована схема циклу $1 \rightarrow \bar{1}$ з БІстабільним керуванням та з одним кінцевим вимикачем

2. Які помилки наведені в схемі на рис. 8.18 ?

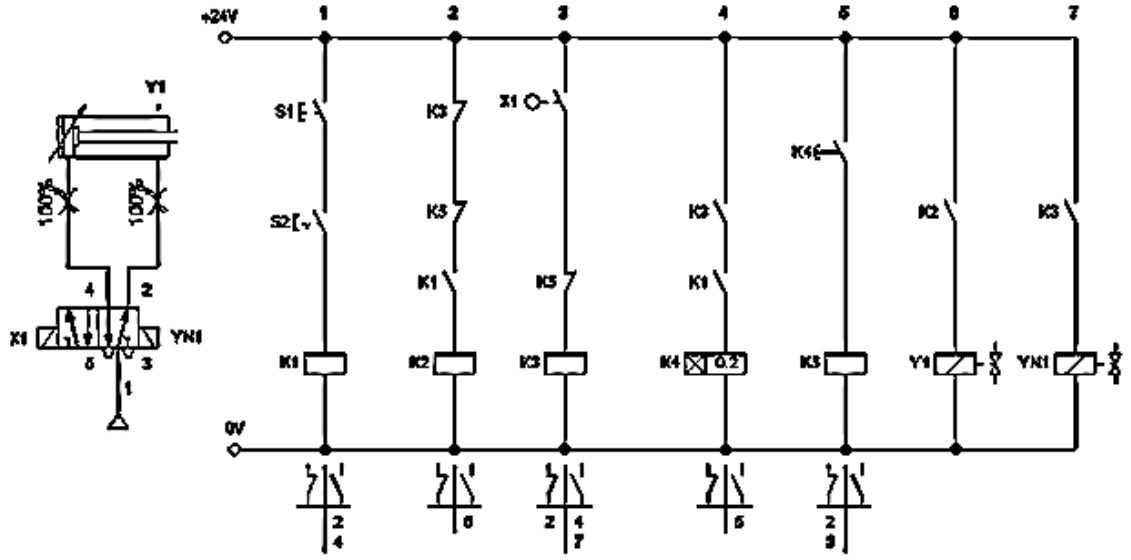


Рис. 8.18. Комбінована схема циклу $1 \rightarrow \bar{1}$ з БІстабільним керуванням та з одним кінцевим вимикачем

3. Для якого такту циклу наведений на рис. 8.19 скриншот анімації роботи схеми ?

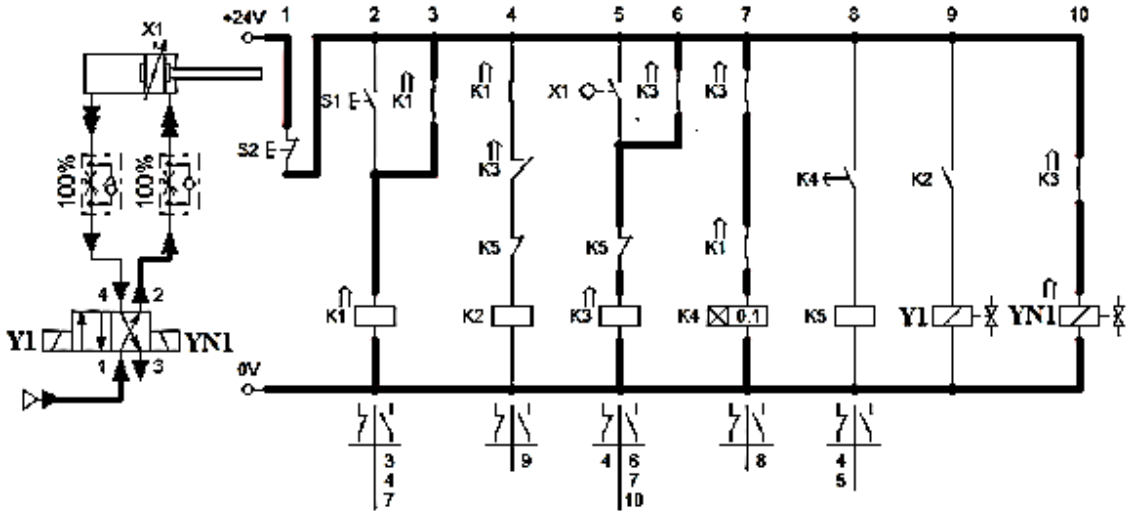


Рис. 8.19. Скриншот анімації циклу «**1–N1**» з БІстабільним керуванням та одним кінцевим вимикачем

8.5. **МОНОстабільне керування виконавчим механізмом з одним кінцевим вимикачем для виконання циклу «1–N1» із затримкою в часі при втягнутому положенні поршня**

В програмному середовищі FluidSim створюємо проект (рис. 8.20)

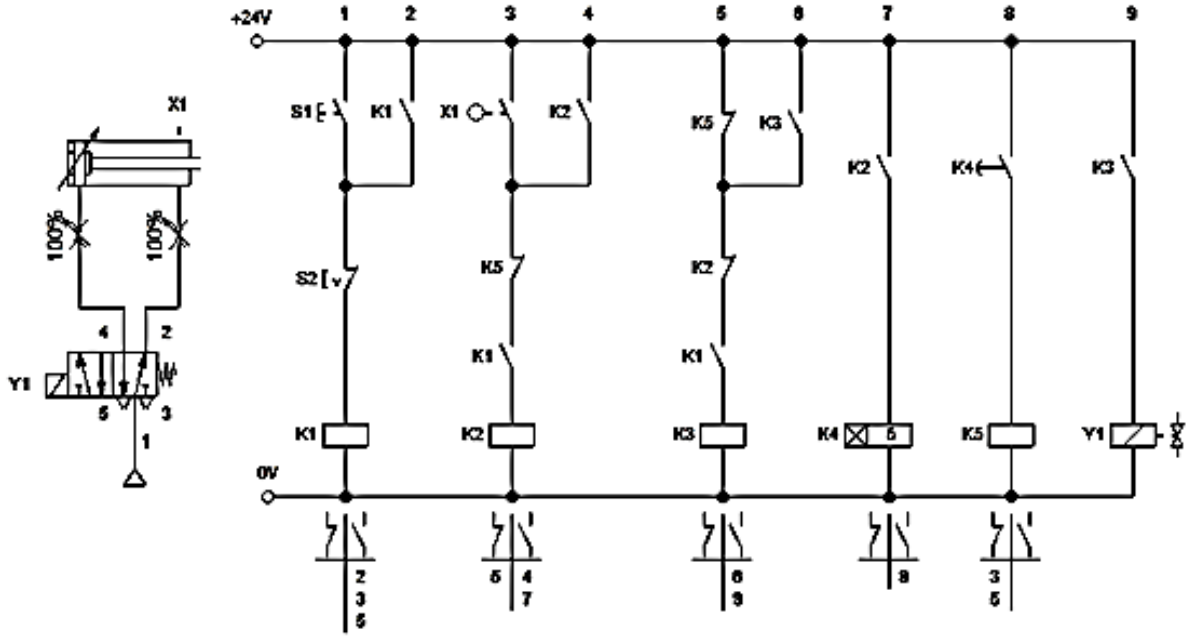


Рис. 8.20. Комбінована схема циклу «1–N1» з МОНОстабільним керуванням з одним кінцевим вимикачем для виконання та затримкою в часі 5 с при втягнутому положенні поршня для реалізації циклу «**1–N1**» з МОНОстабільним керуванням та з одним кінцевим вимикачем

Такий цикл відтворює роботу важільного механізму з програмованою затримкою в часі веденої ланки в одному з крайніх положень.

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.20. треба відповісти на наступні питання.

Чим відрізняється робоча схема на рис. 8.21 від робочої схеми на рис. 8.20 ?

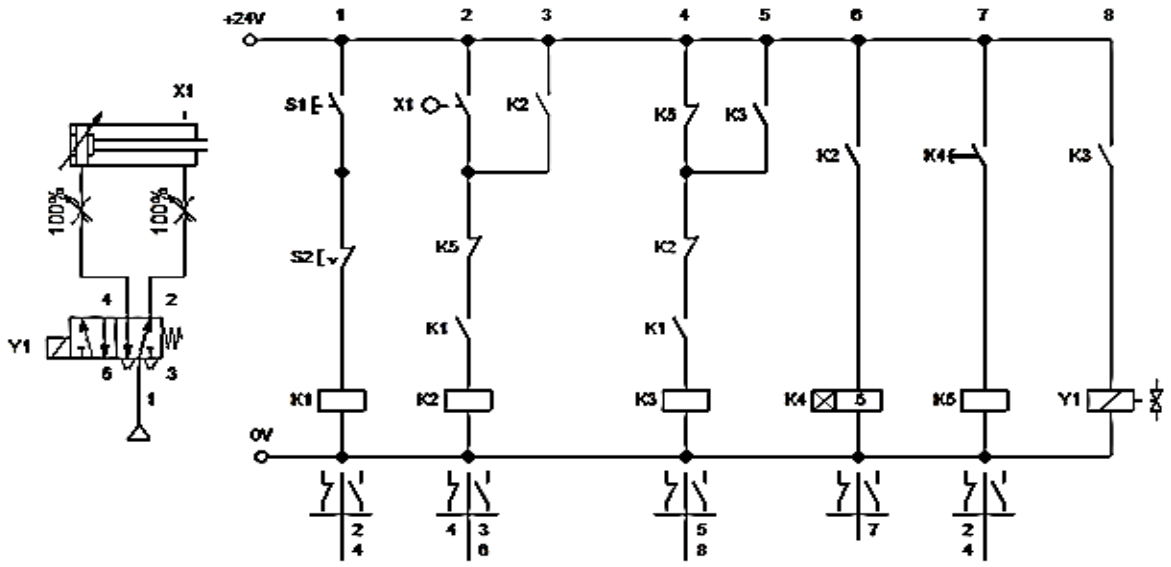


Рис. 8.21. Комбінована схема циклу «1–N1» з МОНОстабільним керуванням з одним кінцевим вимикачем для виконання та затримкою в часі 5 с при втягнутому положенні поршня

Принцип роботи комбінованих схем мехатроніки тут і надалі випливає з наведеної кількості нормально відкритих контактів і нормально закритих контактів К-тих реле, які позначені цифрами у міні таблицях під схемою і ці цифри відповідають цифрам наведених над схемою для відповідних ланцюгів, що містять ці контакти К-тих реле.

Яка помилка в схемі на рис. 8.22 в порівнянні зі схемою на рис. 8.20 ?

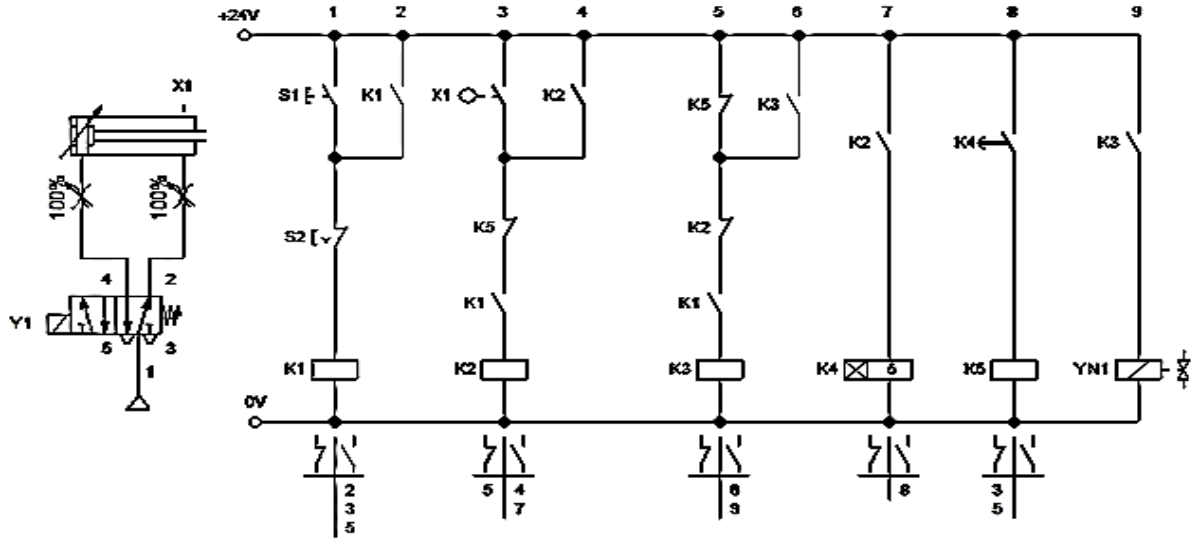


Рис. 8.22. Комбінована схема циклу «1–N1» з МОНОстабільним керуванням з одним кінцевим вимикачем для виконання та затримкою в часі 5 с при втягнутому положенні поршня

Для якого такту циклу наведений на рис.8.23 скриншот анімації циклу роботи схеми?

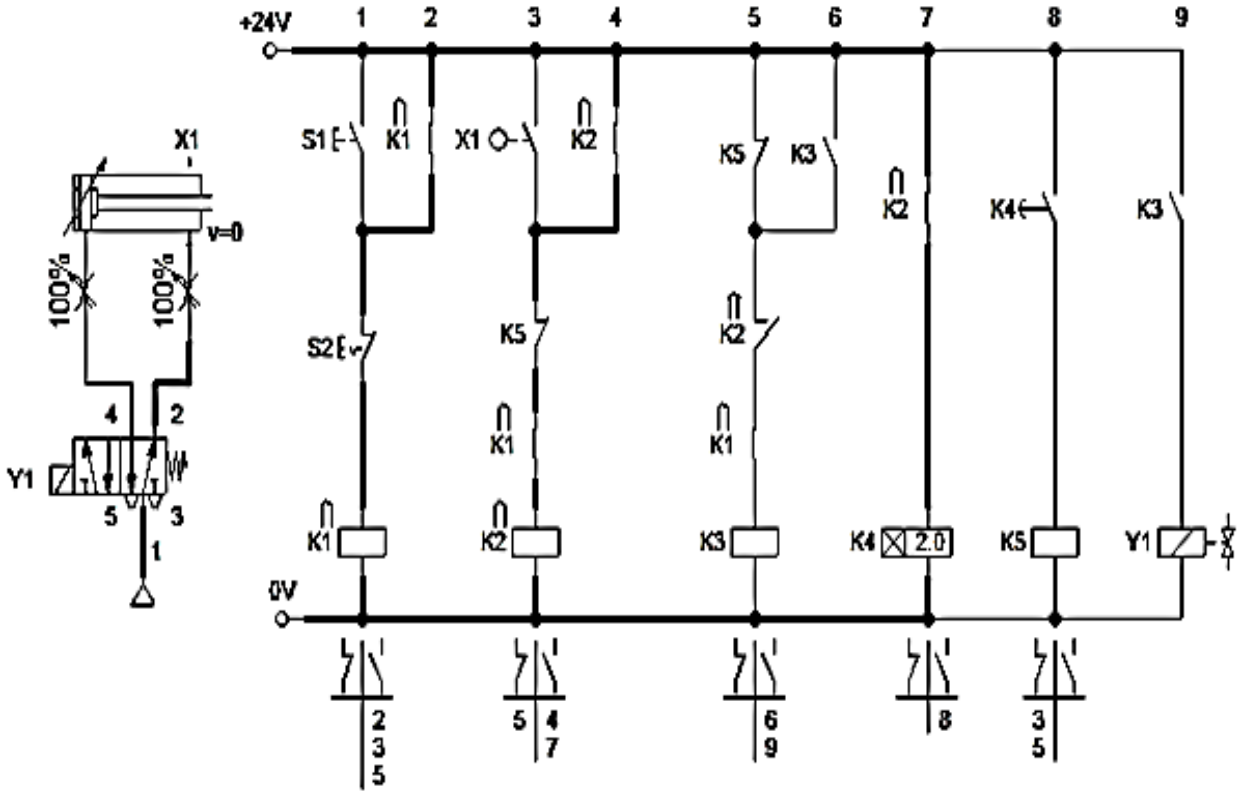


Рис. 8.23. Скриншот анімації циклу «**1–N1**» з МОНОстабільним керуванням та одним кінцевим вимикачем

8.6. **БІстабільне керування виконавчим механізмом без кінцевих вимикачів для виконання циклу «1–N1»**

В програмному середовищі **FluidSim** створюємо проект (рис. 8.24) для реалізації циклу «**1–N1**» з БІстабільним керуванням без кінцевих вимикачів, який відтворює роботу кривошипно-повзунного механізму без кривошипу та без шатуна. і виконуємо анімацію роботи проекту.

На рис.8.24…рис.8.27 наведені комбіновані схеми тип С3 (електрична схема – тип Е3 і пневматична схема – тип П3) для виконання циклу «**1–N1**» з БІстабільним керуванням без кінцевих вимикачів і без контролера.

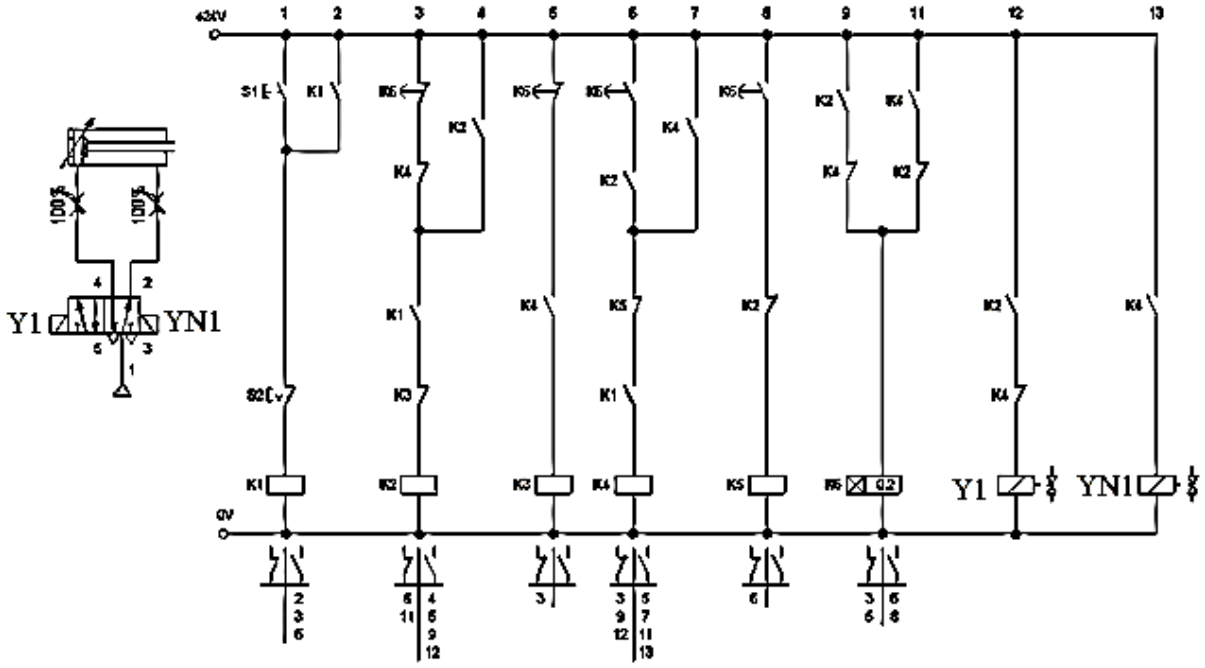


Рис. 8.24. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням без кінцевих вимикачів

Чим відрізняється цикл роботи механізму на рис. 8.25 від циклу роботи механізму на рис. 8.24 ?

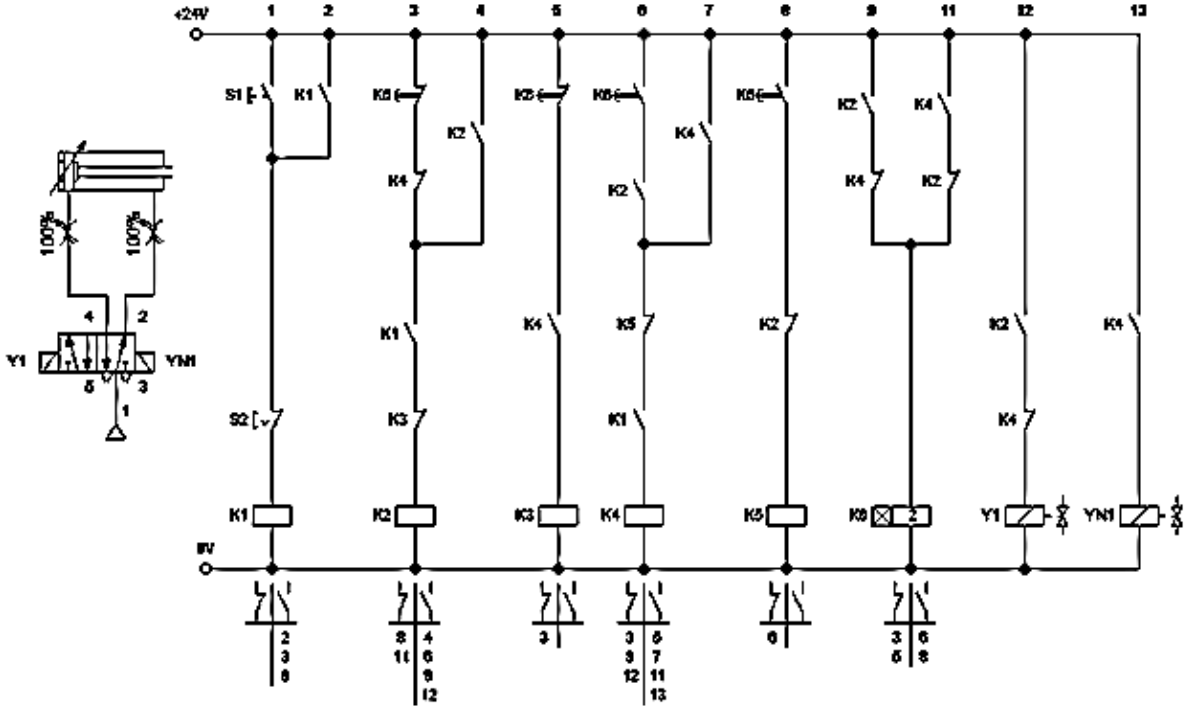


Рис. 8.25. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням без кінцевих вимикачів

Яка помилка в схемі на рис. 8.26 ?

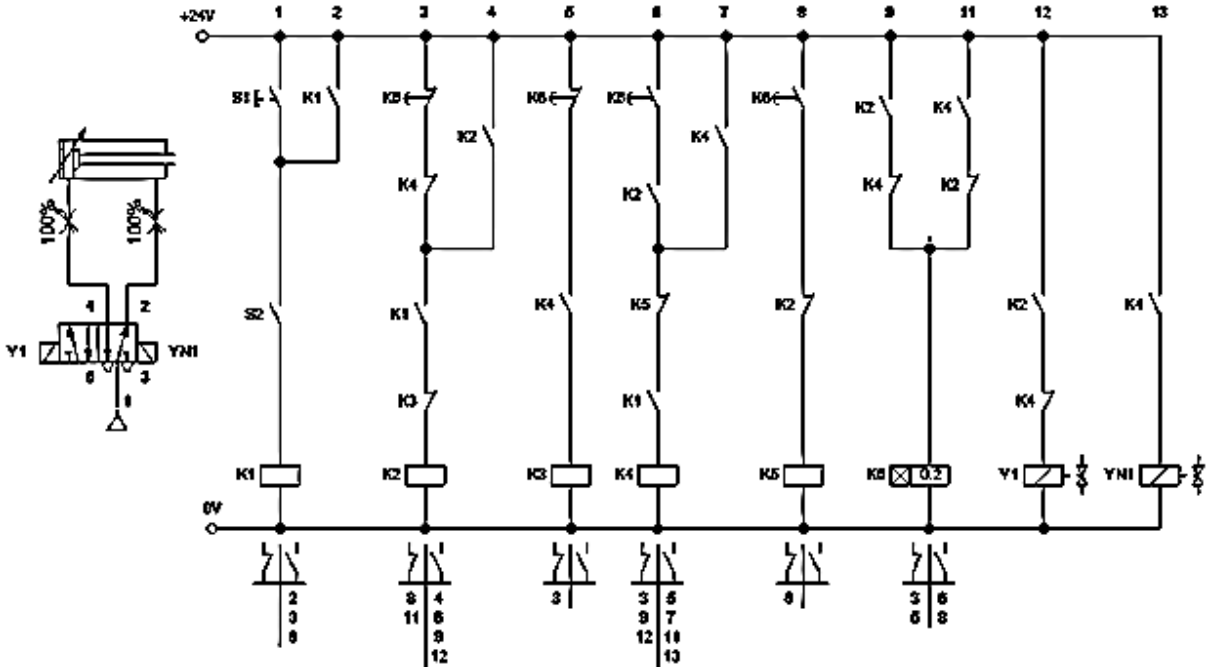


Рис. 8.26. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням без кінцевих вимикачів

Для якого такту циклу наведений на рис. 8.27 скриншот анімації циклу роботи схеми ?

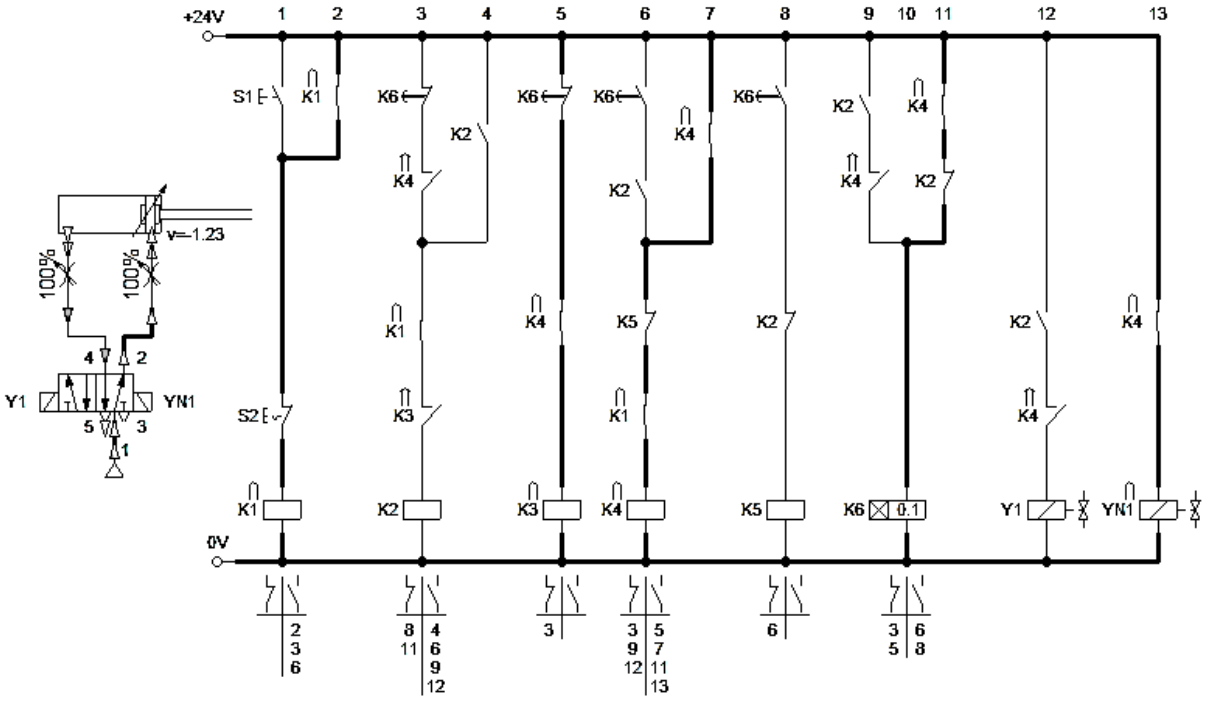


Рис. 8.27. Скриншот анімації циклу «**1–N1**» з БІстабільним керуванням без кінцевих вимикачів

8.7. **МОНОстабільне керування виконавчим механізмом без кінцевих вимикачів для виконання циклу «1–N1»**

В програмному середовищі FluidSim створюємо проект (рис. 8.28).

Реалізації циклу **1– N1** з МОНОстабільним керуванням без кінцевих вимикачів відтворює роботу кривошипно-повзунного механізму без кривошипу та без шатуна.

Принцип роботи схеми на рис. 8.28 і наступних схем пояснюється зміною стану контактів в міні таблицях під електричною схемою після натискання кнопки пуск S1. При виконанні циклу нормально відкриті контакти замикаються, а нормально закриті контакти розмикаються при спрацюванні відповідних реле.

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.28. треба відповісти на наступні питання.

Чим відрізняється цикл роботи механізму на рис. 8.29 від циклу роботи механізму на рис. 8.28 ?

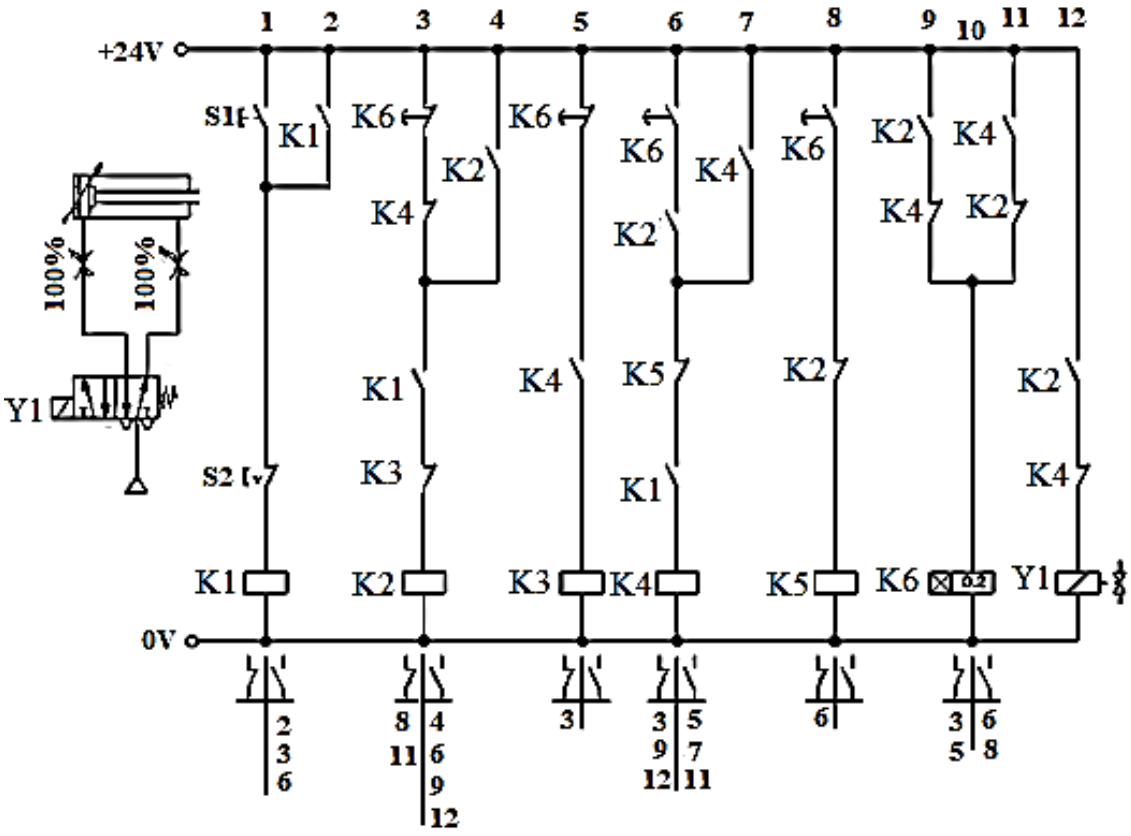


Рис. 8.28. Комбінована схема (тип С3) циклу «**1–N1**» з МОНОстабільним керуванням без кінцевих вимикачів

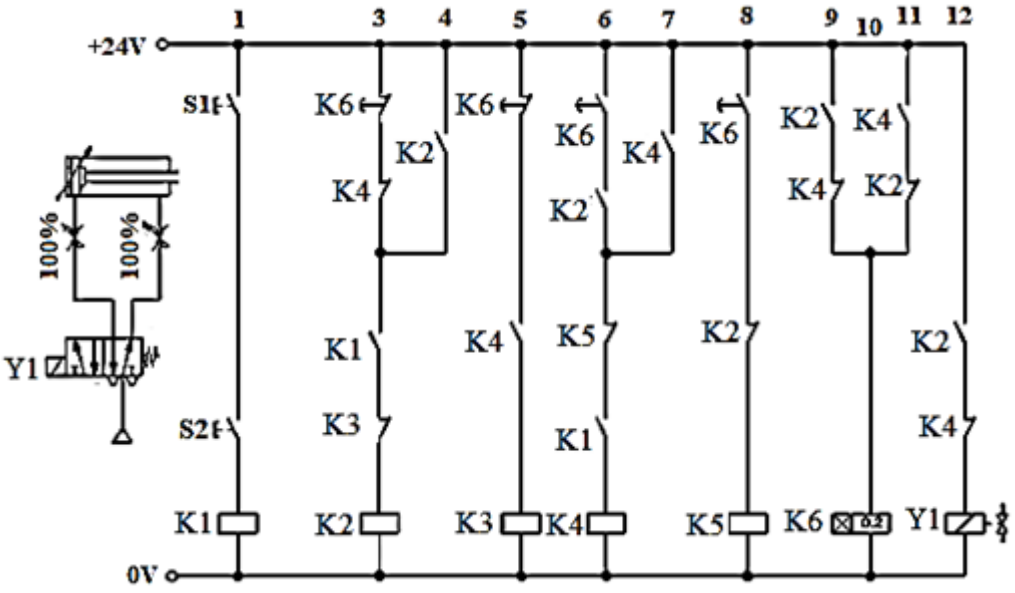


Рис. 8.29. Комбінована схема циклу «**1–N1**» з МОНОстабільним керуванням без кінцевих вимикачів

Яка помилка в схемі на рис.8.30 порівняно зі схемою на рис. 8.28 ?

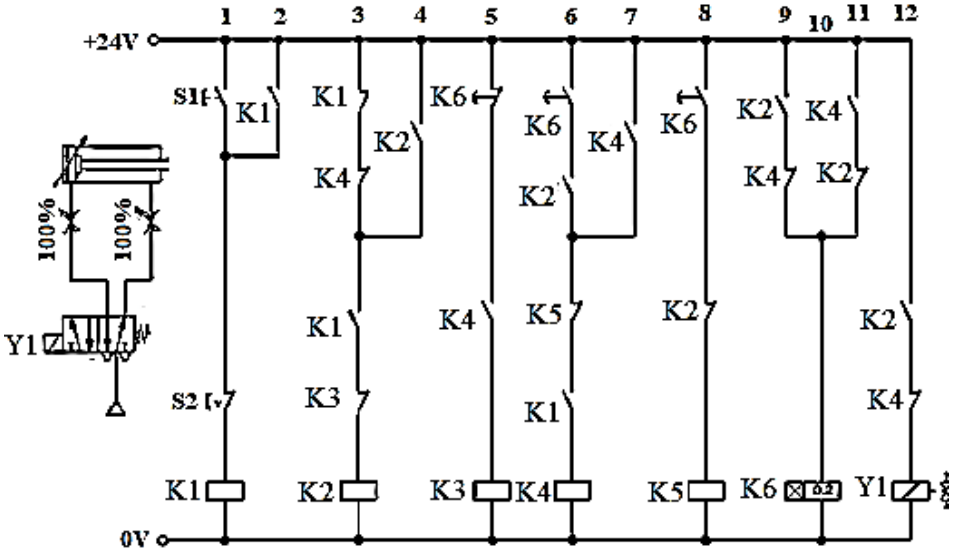


Рис. 8.30. Комбінована схема циклу «**1–N1**» з МОНОстабільним керуванням без кінцевих вимикачів

Для якого такту циклу наведений на рис. 8.31 скриншот анімації циклу роботи схеми ?

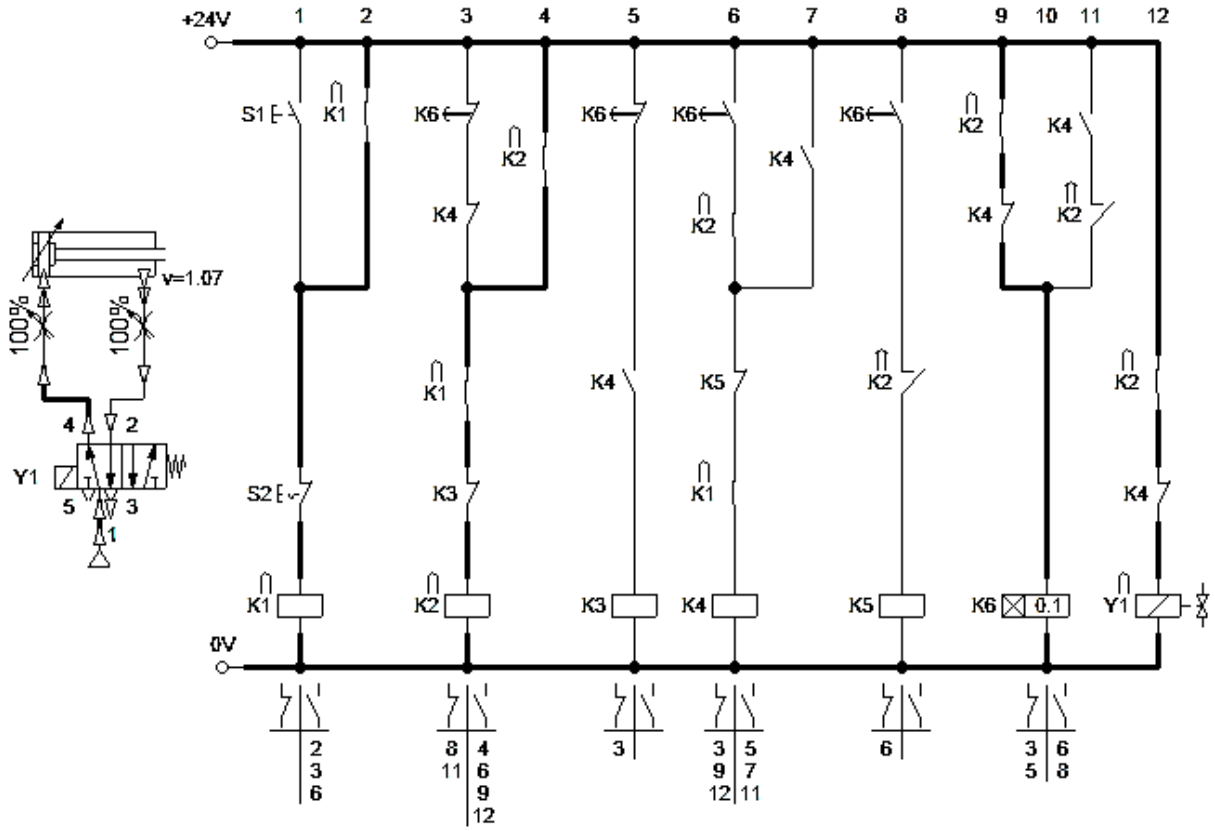


Рис. 8.31. Скриншот анімації циклу «**1–N1**» з МОНОстабільним керуванням без кінцевих вимикачів

8.8. **БІстабільне керування виконавчим механізмом з двома кінцевими вимикачами та реле часу для виконання циклу «1–N1»**

В програмному середовищі FluidSim створюємо проект (рис. 8.32) для реалізації циклу «**1–N1**» з БІстабільним керуванням, двома кінцевими вимикачами та реле часу для затримки зворотного включення пневморозподільника при переміщенні вперед штока пневмоциліндру і виконуємо анімацію роботи проекту.

Принцип роботи схеми на рис. 8.32 і наступних схем пояснюється зміною стану контактів в міні таблицях під електричною схемою після натискання кнопки пуск S1. Нормально відкриті контакти замикаються, а нормапльно закриті контакти розмикаються при спрацюванні відповідних реле.

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.32. треба відповісти на наступні питання.

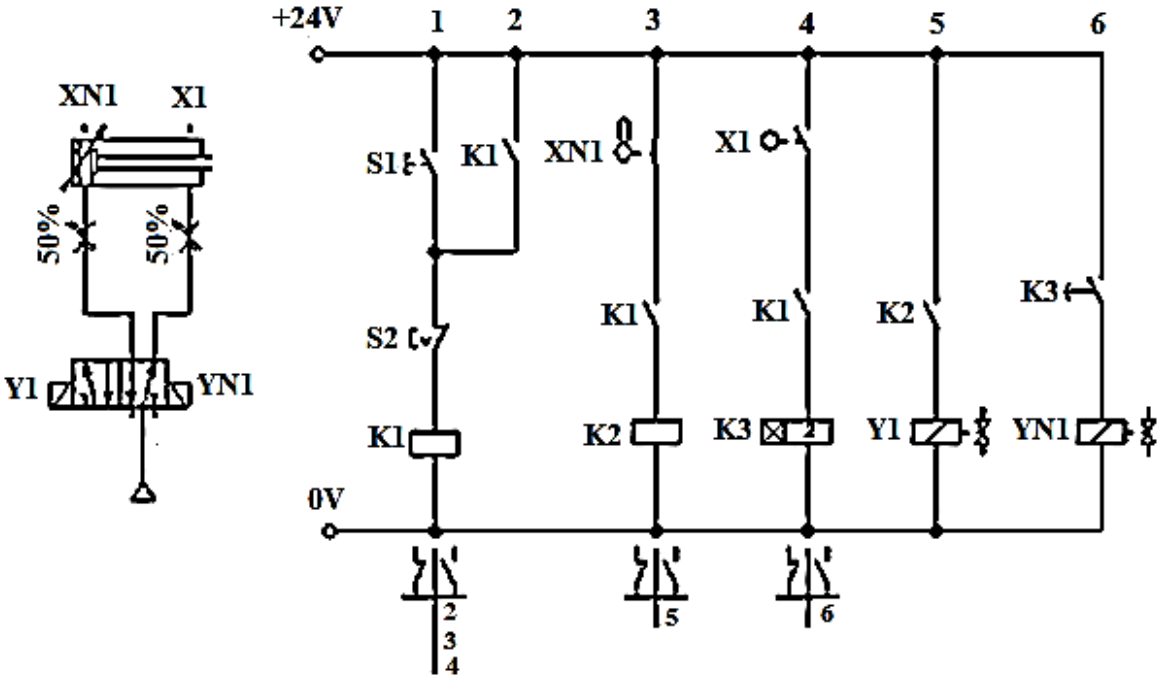


Рис. 8.32. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням, двома кінцевими вимикачами та реле часу для затримки зворотного включення пневморозподільника при переміщенні вперед штока пневмоциліндру

Яка помилка в схемі рис. 8.33 ?

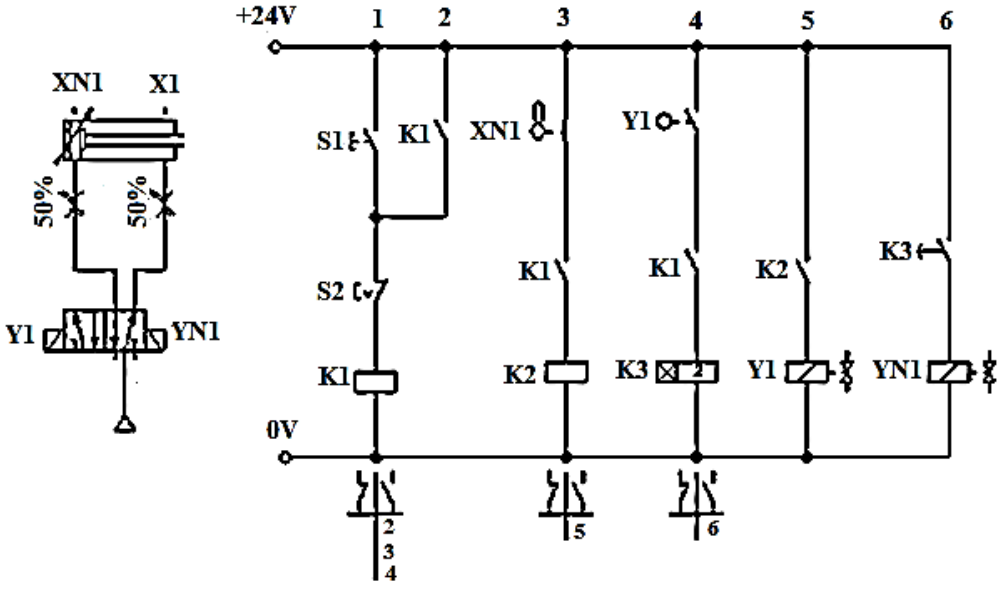


Рис. 8.33. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням, двома кінцевими вимикачами та реле часу

Чим відрізняється схема на рис. 8.34 від схеми на рис. 8.32 ?

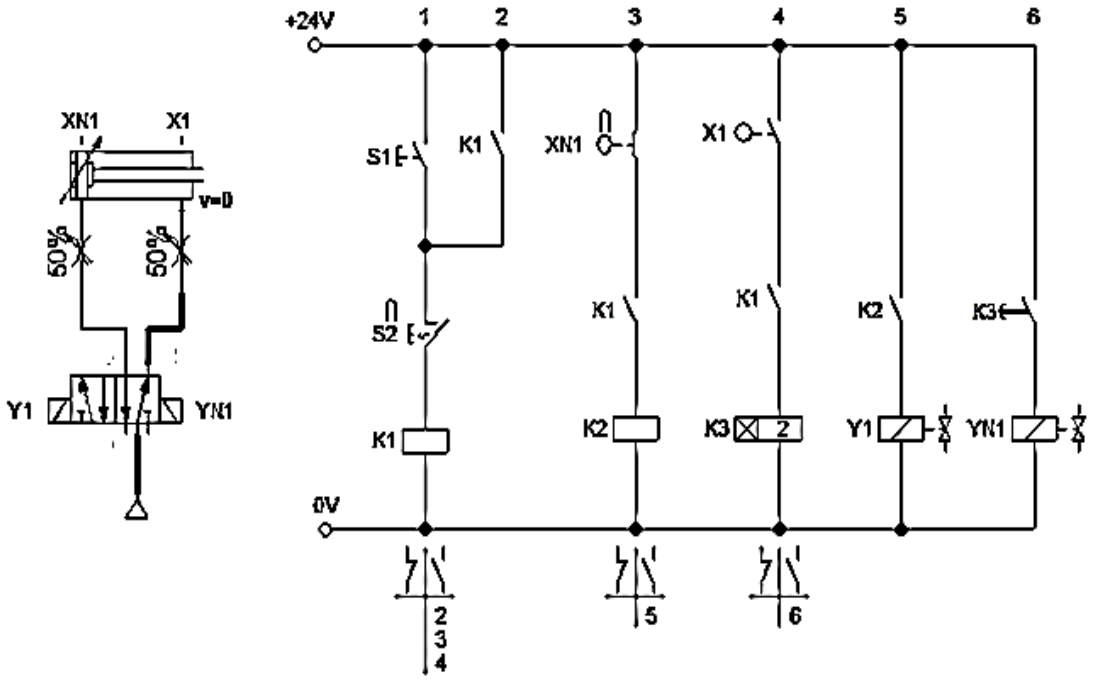


Рис. 8.34. Комбінована схема циклу «**1–N1**» з БІстабільним керуванням, двома кінцевими вимикачами та реле часу

Для якого такту циклу наведений на рис. 8.35 скриншот анімації циклу роботи схеми ?

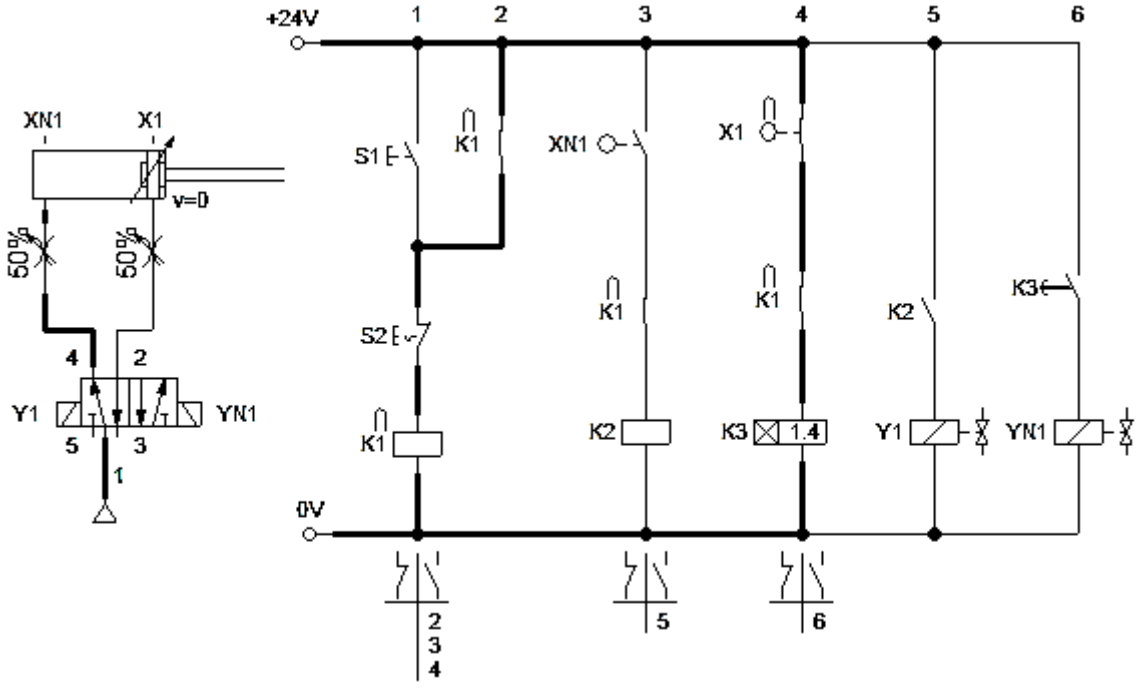


Рис. 8.35. Скриншот анімації циклу «**1–N1**» з БІстабільним керуванням з двома кінцевими вимикачами та реле часу

8.9. **БІстабільне керування двома виконавчими механізмами з чотирма кінцевими вимикачами для виконання циклу $\mathbf{1 \to 2 \to \overline{1} \to \overline{2}}$**

Розглянемо приклад складання логічних команд для функціонального модуля з БІстабільним керуванням і використанням методу графів. Граф це множина, яка складається з підмножини вершин (позначені на графі рис. 8.36 точками) та підмножини ребр (позначені на графі рис. 8.36 стрілками-дугами). Початок кожній стрілки це стан сигналу (кнопки, кінцевого вимикача), а той же стрілки – необхідна команда на включення (пряма команда) та вимкнення (зворотна команда) електромагніту пневморозподільника або включення/вимкнення відповідного елементу пам'яті, таймера та лічильника.

Нехай необхідно створюємо систему керування трьома пневматичними циліндрами з БІстабільним керуванням, які працюють по наступному циклу $\mathbf{1 \to 2 \to 3 \to \overline{1} \to \overline{2} \to \overline{3}}$. Команди для такого циклу будемо складати за допомогою методу графів. Для цього креслимо коло і на нього наносимо послідовність сигналів заданого циклу $\mathbf{1 \to 2 \to 3 \to \overline{1} \to \overline{2} \to \overline{3}}$ (вершини графу) і з'єднуємо протилежні вершини лініями зв'язку (рис. 8.36).

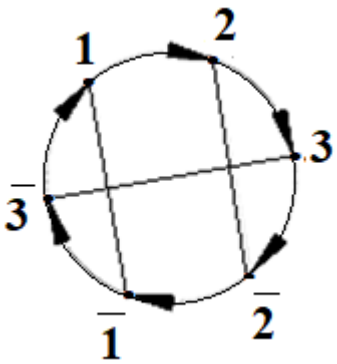


Рис. 8.36. Граф циклу роботи системи

Для запису команди керування необхідно зайти з кінця попередньої дугі відносно необхідної команди в початок наступної по лініям зв'язку, записуючи всі сигнали. Складемо пряму команду Y_1 для включення електромагніту пневморозпордільника першого пневмоциліндру:

$$\boldsymbol{Y_1 \Leftarrow X_{\overline{3}} \cdot X_{\overline{1}}}\ . \qquad (8.10)$$

Як бачимо в кінці команди Y_1 присутній сигнал $X_{\bar{1}}$. Його можна не записувати, оскільки зрозуміло, що для того, щоб шток виїхав він повинен спочатку знаходитись у втягнутому положенні.

Тому при написанні БІстабільних команд керування останній сигнал можна не враховувати. Але в першій команді циклу необхідно врахувати сигнал від стартової кнопки X_{st}. Команда Y_1 записується у лівої частині рівняння (умови) причинно-наслідкових зв'язків, а права частина утворюються станом необхідних сигналів для реалізації прямий команди :

$$\boldsymbol{Y_1 \Leftarrow X_{\bar{3}} \cdot X_{ST}} . \qquad (8.11)$$

Цю команду можна прочитати наступним чином: «Пряма команда для першого виконавчого механізму (пневмоциліндру) буде реалізована, якщо поршень третього пневмоциліндру буде втягнутий і при цьому буде замкнений кінцевий вимикач XN3 та натиснута кнопка старту X_{st}».

Аналогічно записуємо всі інші команди у відповідності до фрагментів графу на рис. 8.37.

$$\boldsymbol{Y_{\bar{1}} \Leftarrow X_{\bar{2}} \cdot X_{\bar{3}}} ; \qquad (8.12)$$

$$\boldsymbol{Y_2 \Leftarrow X_1 \cdot X_{\bar{3}}} ; \qquad (8.13)$$

$$\boldsymbol{Y_{\bar{2}} \Leftarrow X_3} ; \qquad (8.14)$$

$$\boldsymbol{Y_3 \Leftarrow X_2} ; \qquad (8.15)$$

$$\boldsymbol{Y_{\bar{3}} \Leftarrow X_{\bar{1}}} . \qquad (8.16)$$

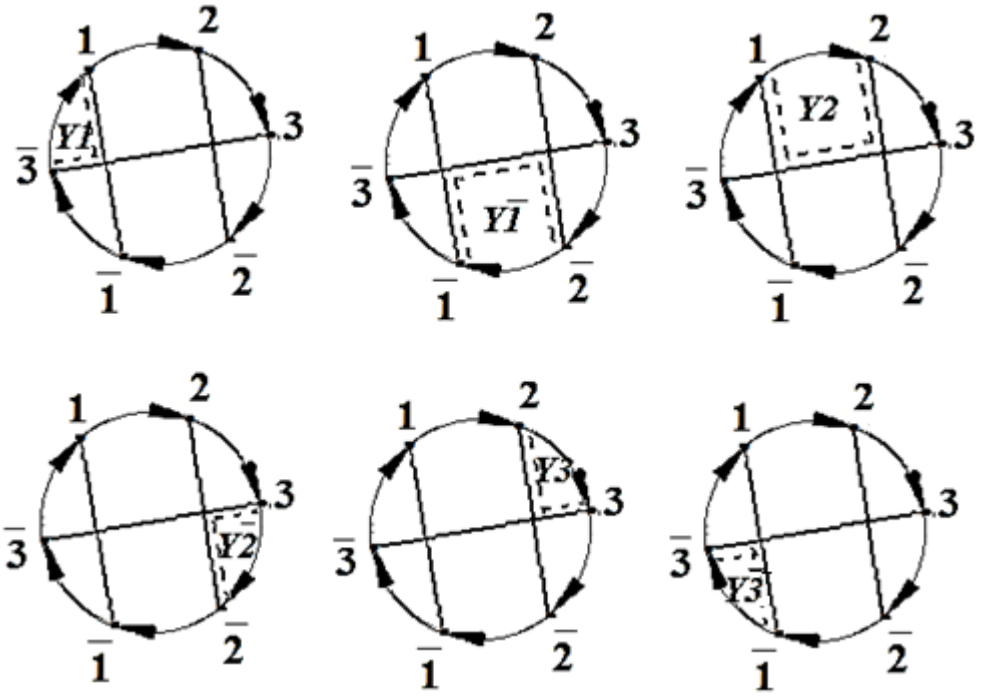


Рис. 8.37. Приклад складання БІстабільних команд керування для циклу $\mathbf{1 \to 2 \to 3 \to \overline{1} \to \overline{2} \to \overline{3}}$

В програмному середовищі FluidSim створюємо проект (рис. 8.38) для реалізації циклу $\mathbf{1 \to 2 \to \overline{1} \to \overline{2}}$ з БІстабільним керуванням двома виконавчими механізмами та чотирма кінцевими вимикачами виконуємо анімацію роботи проекту. Складаємо функціональний граф і систему рівнянь причино-наслідкових зв'язків для складання схеми на рис. 8.38.

На лабораторному стенді необхідно зібрати схему (рис. 8.38) для реалізації циклу по п.1.

Принцип роботи схеми на рис. 8.38 і наступних схем пояснюється зміною стану контактів в міні таблицях під електричною схемою після натискання кнопки пуск S1. Нормально відкриті контакти замикаються, а нормапльно закриті контакти розмикаються при спрацюванні відповідних реле.

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.32. треба відповісти на наступні питання.

1. Чим відрізняється схема, яка наведена на рис. 8.39 від схеми на рис. 8.38 ?

2. Яке призначення двох кнопок S0, S1 і S2 на рис. 8.38 ?

3. Що потрібно зробити в схемі на рис. 8.38 для відпрацювання

тільки одного циклу роботи мехатронної системи ?

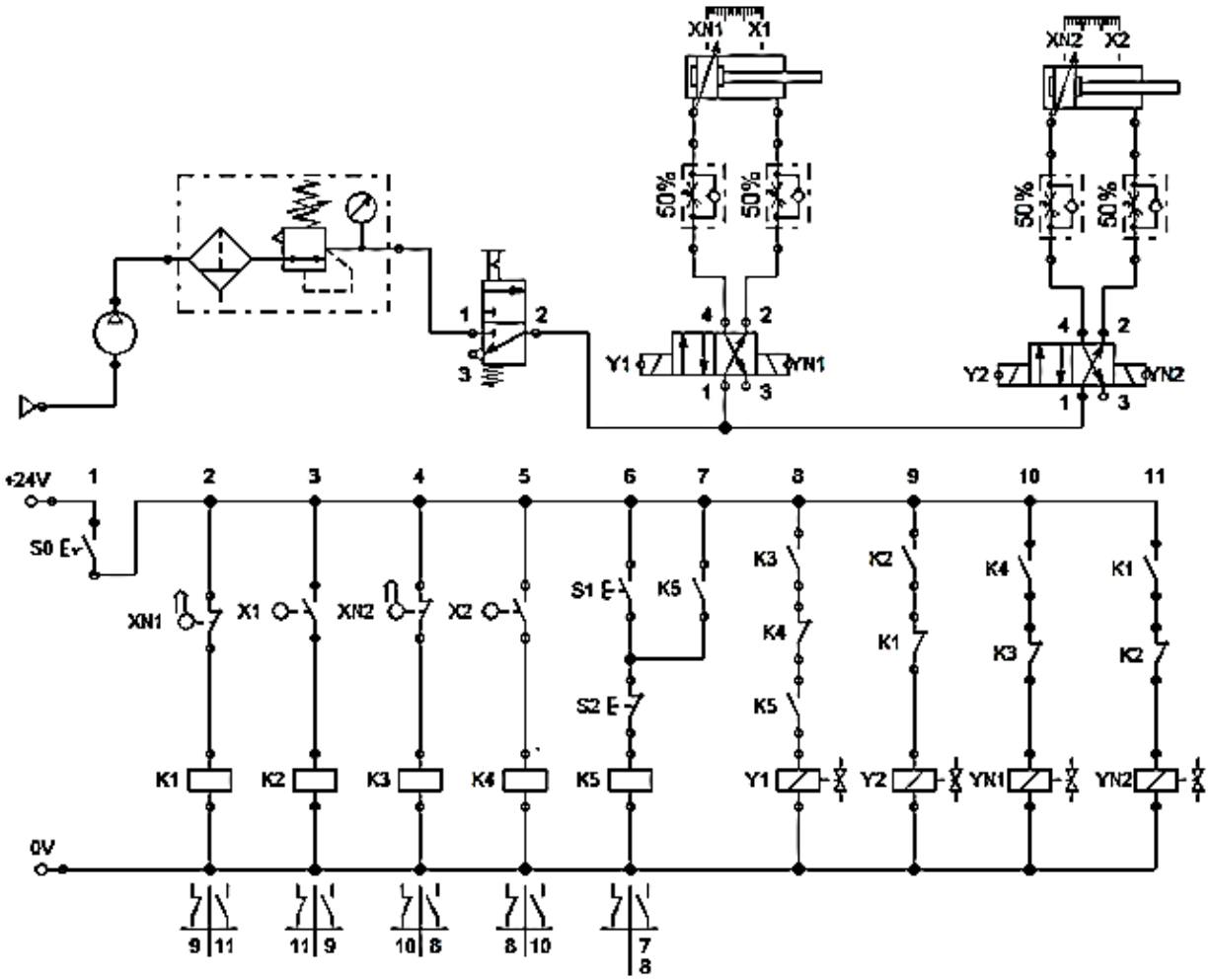


Рис. 8. 38. Комбінована схема циклу $\boxed{1 \to 2 \to \bar{1} \to \bar{2}}$ з БІстабільним керуванням двома виконавчими механізмами

4.Що потрібно зробити в схемі на рис. 8.38 для відпрацювання циклу $1 \to 2 \to \bar{1}\,, \bar{2}$ роботи мехатронної системи ?

5. Що потрібно зробити в схемі на рис. 8.38 для відпрацювання циклу $1 \to \bar{1}\,, 2 \to \bar{2}$ роботи мехатронної системи ?

6. Запишіть БІстабільні команди для циклу $\mathbf{1 \to 2 \to 3 \to \bar{1} \to \bar{3} \to \bar{2}}$ використовуючи метод графів

7. Який порядок включення комбінованої схеми на рис. 8.38 ?

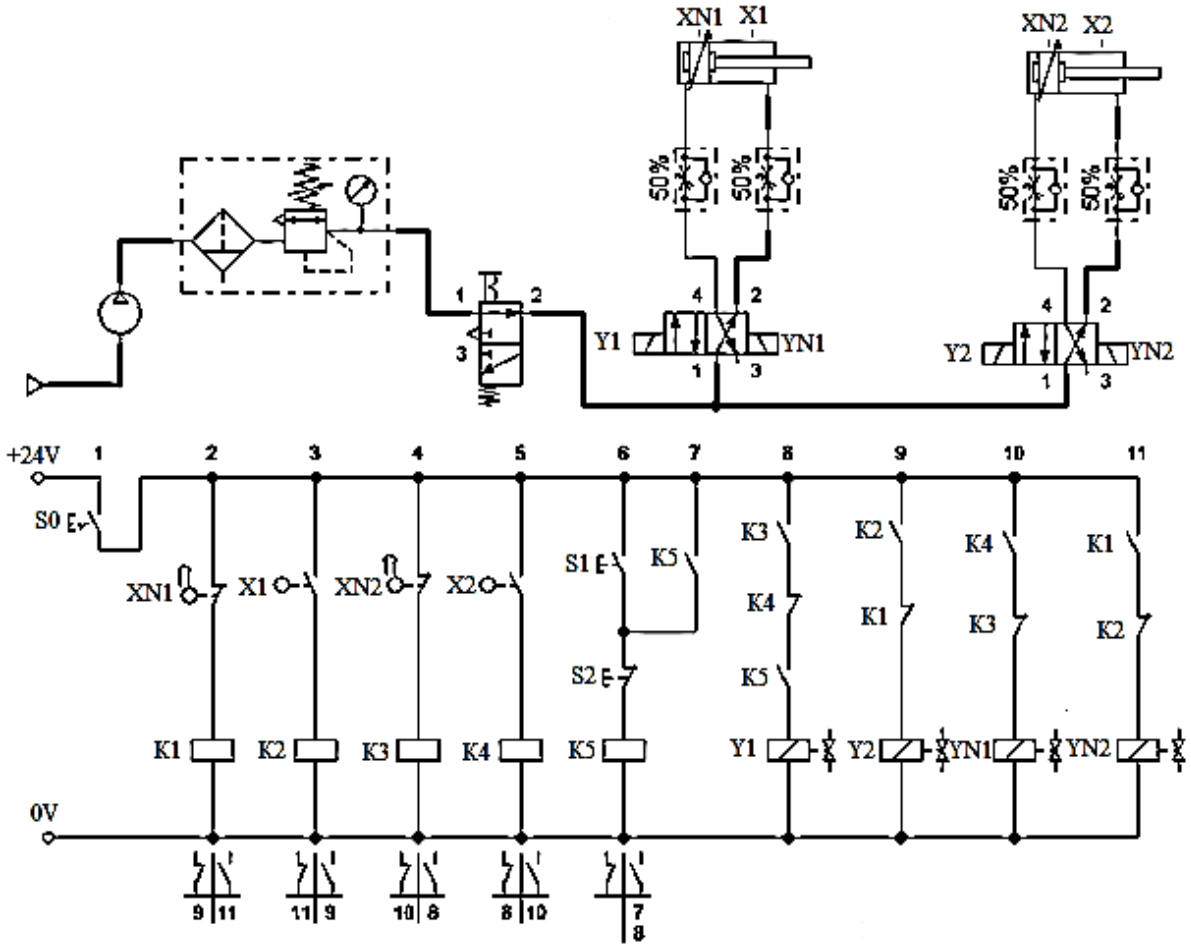


Рис. 8.39. Комбінована схема циклу $1 \to 2 \to \overline{1} \to \overline{2}$ з БІстабільним керуванням двома виконавчими механізмами

.

8.10. **МОНОстабільне керування двома виконавчими механізмами з чотирма кінцевими вимикачами для виконання циклу $1 \to 2 \to \overline{1} \to \overline{2}$**

В промисловості часто необхідно забезпечити повернення приводів в початкове положення у випадку аварійних ситуацій, таких як збій подачі живлення на електричну частину системи. В такому випадку використовують пневморозподільники з МОНОстабільним керуванням, які змінюють позицію при подачі вхідного сигналу, а при його знятті повертаються в початкове положення.

Алгоритм складання команд при МОНОстабільному керуванні відрізняється від складання команд для БІстабільного керування. Нехай необхідно створюємо систему керування трьома пневматичними циліндрами з МОНОстабільним керуванням, які працюють по наступному

циклу $\mathbf{1 \to 2 \to 3 \to \overline{1} \to \overline{2} \to \overline{3}}$. Команди для такого досить складного циклу краще складати за допомогою методу графів. Для цього креслимо граф циклу, на нього наносимо послідовність сигналів $\mathbf{1 \to 2 \to 3 \to \overline{1} \to \overline{2} \to \overline{3}}$ і з'єднуємо протилежні вершини лініями (лініями зв'язку) (рис. 8.40).

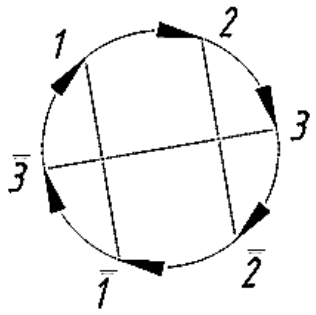


Рис. 8.40. Граф циклу $\mathbf{1 \to 2 \to 3 \to \overline{1} \to \overline{2} \to \overline{3}}$ роботи системи

Для запису команди необхідно зайти з кінця попередньої дуги відносно необхідної команди по лініям зв'язку в кінець попередньої дуги відносно команди інверсної до необхідної (рис. 8.46). У випадку, якщо лінії зв'язку утворюють гострий чи прямий кут зі сторони необхідної дуги, то між сигналами ставиться знак логічного множення «·». Якщо кут між лініями зв'язку тупий – ставиться знак логічного додавання «+». Для останнього сигналу обов'язково робиться інверсія сигналу та індексу.

Складаємо команди для циклу (рис. 8.41). В першу команду додаємо сигнал кнопки «Старт» S1.

$$\boldsymbol{Y_1 \Leftarrow S1 \cdot X_{\overline{3}} \cdot \overline{X}_{\overline{2}};} \qquad (8.17)$$

$$\boldsymbol{Y_2 \Leftarrow X_1 \cdot \overline{X}_3;} \qquad (8.18)$$

$$\boldsymbol{Y_3 \Leftarrow X_2 + X_3 \cdot \overline{X}_{\overline{1}}.} \qquad (8.19)$$

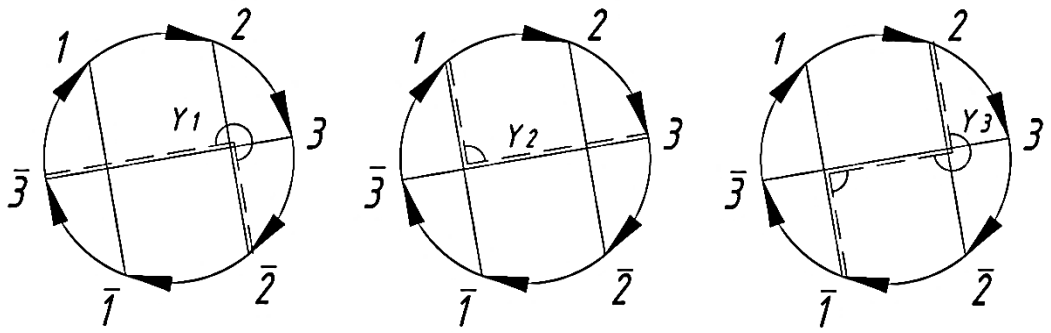


Рис. 8.41. Приклад складання МОНОстабільних команд керування

Для реалізації інверсного сигналу в електричній схемі використовується нормально розімкнений контакт реле.

При МОНОстабільному керуванні приводом з розподільником 5/2 складається тільки одна команда, а коли вона не виконується розподільник автоматично повертається в початкове положення і пневмопривод також переходить в початкове положення.

Окремим випадком складання МОНОстабільних команд є запис команди для першого привода в циклі $\mathbf{1 \to 2 \to \overline{1} \to \overline{2}}$ (рис. 8.42). Для першого привода команда складається наступним чином:

$$\boldsymbol{Y_1 \Leftarrow X_{\overline{2}} + X_1 \cdot \overline{X}_2} \qquad (8.20)$$

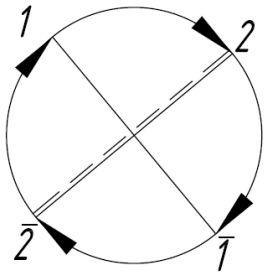


Рис. 8.42. Граф циклу $\mathbf{1 \to 2 \to \overline{1} \to \overline{2}}$ роботи системи

В програмному середовищі **FluidSim** створюємо проект для реалізації циклу $1 \to 2 \to \overline{1} \to \overline{2}$ з БІстабільним керуванням двома виконавчими механізмами і виконуємо анімацію роботи проекту (рис. 8.43).

У відповідності до функціонального графу на рис. 8.42 і рівняння причино-наслідкових зв'язків (1) складаємо схему, наведену на рис. 8.43.

Принцип роботи схеми на рис. 8.43 і наступних схем пояснюється зміною стану контактів в міні таблицях під електричною схемою після натискання кнопки пуск S1. Нормально відкриті контакти замикаються, а нормально закриті контакти розмикаються при спрацюванні відповідних реле.

Для закріплення матеріалу стосовно схеми мехатроніки на рис. 8.48 треба відповісти на наступні питання:

1. Чим відрізняється схема на рис. 8.44 від схеми на рис. 8.43 ?
2. Для циклу $\mathbf{1 \to 2 \to 3 \to \overline{2} \to \overline{1} \to \overline{3}}$ скласти МОНОстабільні зворотні команди.

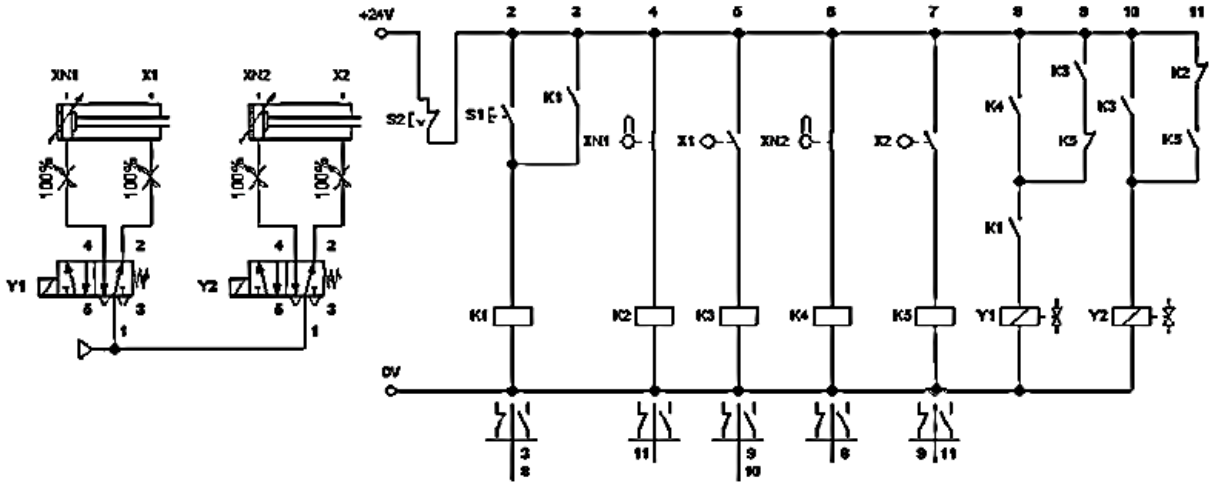


Рис. 8.43. Комбінована схема циклу (тип С3) $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ з МОНОстабільним керуванням та чотирма кінцевими вимикачами

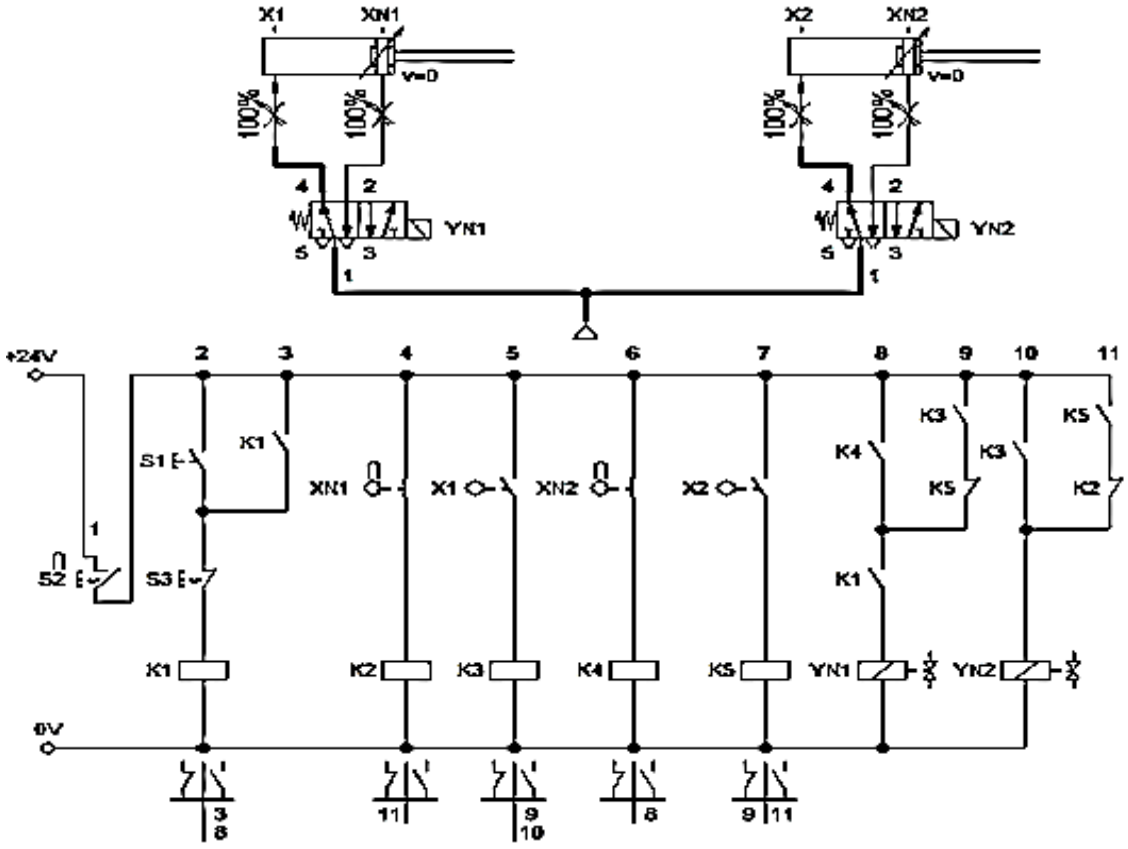


Рис. 8.44. Комбінована схема циклу $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ з МОНОстабільним керуванням з чотирма кінцевими вимикачами

8.11. **БІстабільне керування двома виконавчими механізмами з чотирма кінцевими вимикачами для циклу $\mathbf{1 \to \overline{1} \to 2 \to \overline{2}}$**

Розглянемо цикл роботи системи з двома пневмоциліндрами, які виконують роботу по циклу $\mathbf{1 \to \overline{1} \to 2 \to \overline{2}}$. В такій системі підчас її роботи один і той же стан повторюється двічі (на початку і після відпрацювання першого пневмоциліндру обидва штоки втягнуті). В такому стані система «не знає» чи починає вона цикл, чи вже відпрацювала половину циклу. Побудуємо граф циклу (рис. 8.45).

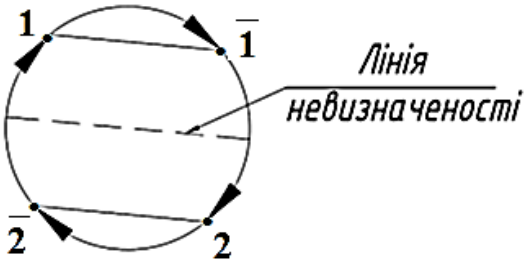


Рис. 8.45. Граф циклу системи з лінією невизначеності

З графу циклу на рис. 8.46випливає, що лінії зв'язку приводів 1 та 2 не перетинаються. Такий граф розпадається на два підграфи невизначеності, а між ними проходить **лінія невизначеності**. Взагалі, лінія невизначеності проводиться так, що вона:

-не перетинає жодної лінії зв'язку;

-ділить граф на дві частини, в кожній з яких є хоча б одна лінія зв'язку.

Якщо в системі є хоча б одна лінія невизначеності, то вона називається нереалізованою.

Для того, щоб система стала реалізованою її необхідно доповнити додатковим функціональним модулем - **елементом пам'яті**. Елемент пам'яті вводиться в систему таким, щоб його лінія зв'язку перетинала якомога більшу кількість ліній невизначеності, а також, по можливості, ліній зв'язку інших одиниць системи. Елемент пам'яті вводиться в цикл системи таким чином, щоб в початковий момент роботи системи всі функціональні модулі системи перебували в початковому положенні.

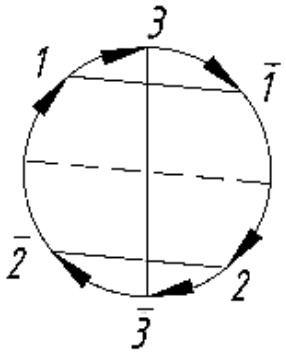
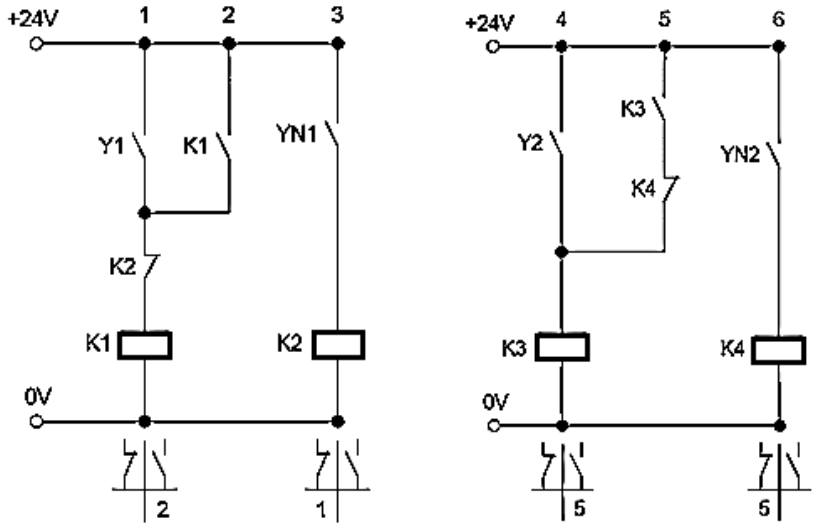


Рис. 8.46. Граф циклу після додавання елементу пам'яті **3**

Отже, в наведеному прикладі лінія невизначеності прибирається додатковим елементом пам’яті 3. Граф циклу вигляду згідно з рис. 8.46) набуває наступного вигляду :

$$\mathbf{1} \rightarrow \mathbf{3} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{2} \rightarrow \overline{\mathbf{3}} \rightarrow \overline{\mathbf{2}}$$

Для елемента пам’яті записуються БІстабільні команди. В пневматиці елемент пам’яті реалізується БІстабільним пневморозподільником 5/2 з пневматичним керуванням. В електричних схемах **елемент пам’яті** реалізується за допомогою реле та їх контактів. Розповсюдженими є елементи пам’яті з пріоритетом на та вимикання та вмикання (рис. 8.47).



а б

Рис. 8.47. Елементи пам’яті на електромагнітних реле: а – з пріоритетом на вмикання; б – з пріоритетом на вимикання

В програмному середовищі **FluidSim** створюємо проект для реалізації циклу $\mathbf{1} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{2} \rightarrow \overline{\mathbf{2}}$ з БІстабільним керуванням двома виконавчими механізмами з чотирма кінцевими вимикачами виконуємо анімацію роботи схеми (рис. 8.43). В пневматичної схемі використана двопозиційна пневмокнопка з фіксацією у нетисненому стані.

Згідно з функціональним графом на рис. 8.46 складаємо систему рівнянь причино-наслідкових зв’язків, які використовуються для складання схеми на рис. 8.48.

$$\boldsymbol{Y_1 \Leftarrow X_{\overline{2}} \cdot X_{\overline{3}} \cdot S_2}\ ; \qquad (8.21)$$

$$Y_{\bar{1}} \Leftarrow X_3\,; \tag{8.22}$$

$$Y_2 \Leftarrow X_{\bar{1}} \cdot X_3\,; \tag{8.23}$$

$$Y_{\bar{2}} \Leftarrow X_{\bar{3}}\,; \tag{8.24}$$

$$Y_3 \Leftarrow X_1\,; \tag{8.25}$$

$$Y_{\bar{3}} \Leftarrow X_2\,. \tag{8.26}$$

Принцип роботи схеми на рис. 8.48 і наступних схем пояснюється зміною стану контактів в міні таблицях під електричною схемою після натискання кнопки пуск S1. Нормально відкриті контакти замикаються, а нормапльно закриті контакти розмикаються при спрацюванні відповідних реле.

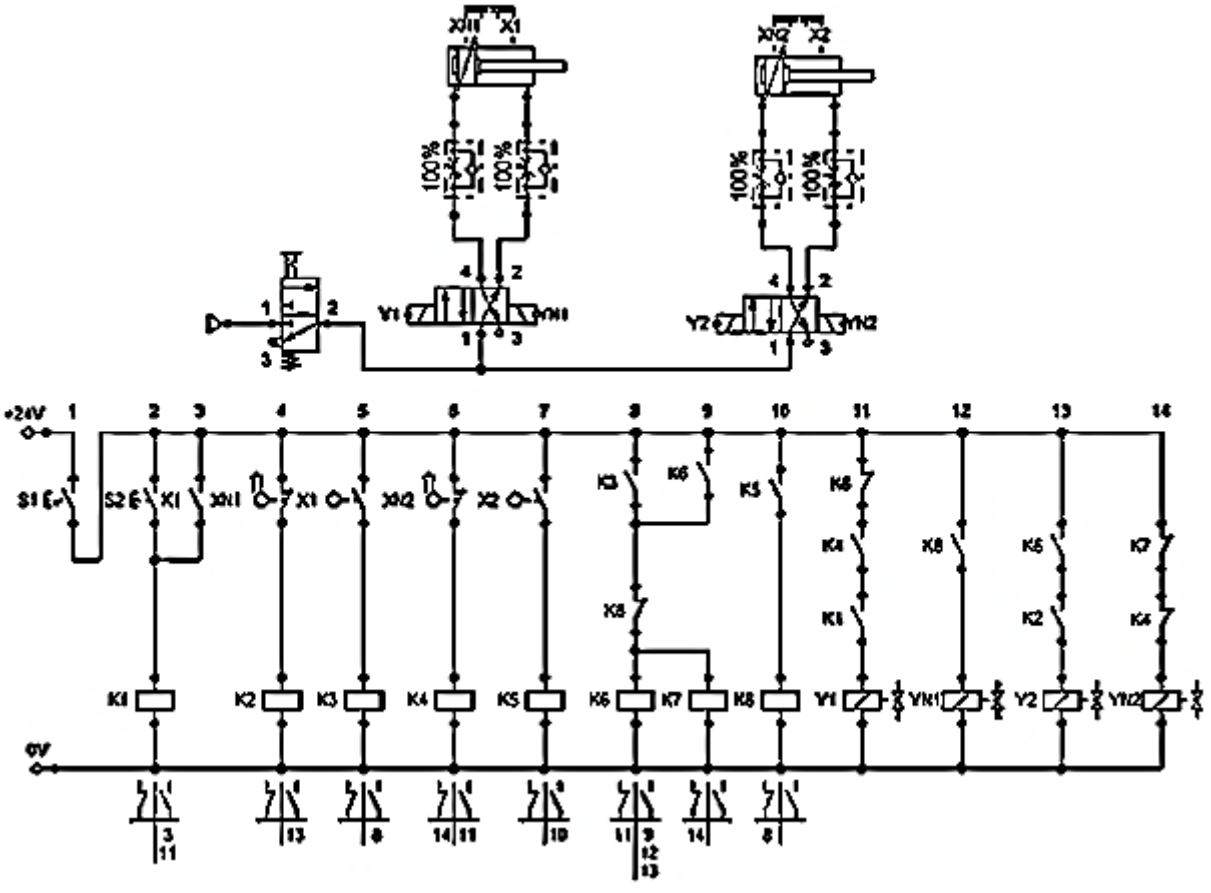


Рис. 8.48. Комбінована схема циклу $\mathbf{1 \to \bar{1} \to 2 \to \bar{2}}$ з БІстабільним керуванням двома виконавчими механізмами

На рис. 8.49 наведена комбінована схема (тип С3) циклу $1 \to \bar{1} \to 2 \to \bar{2}$ при включеному стані пневматичної схеми (тип П3) і вимкненого

стані електричної схеми (тип Е3), а на рис. 8.50 – комбінована схема при вимкненому стані пневматичної схеми і включеному стані електричної схеми.

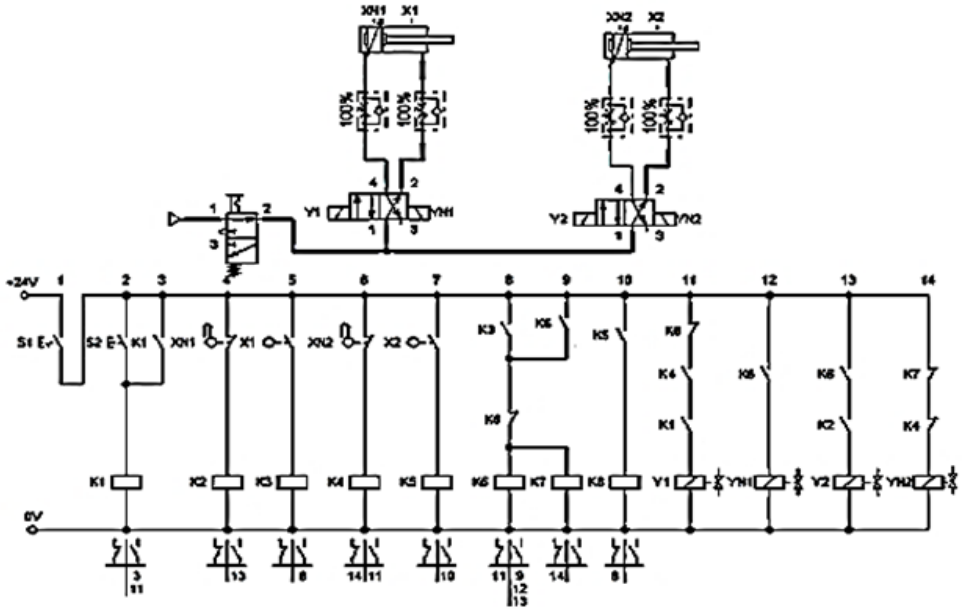


Рис. 8.49. Комбінована схема циклу $\mathbf{1 \rightarrow \overline{1} \rightarrow 2 \rightarrow \overline{2}}$ при включеному стані пневматичної схеми і вимкненого стані електричної схеми

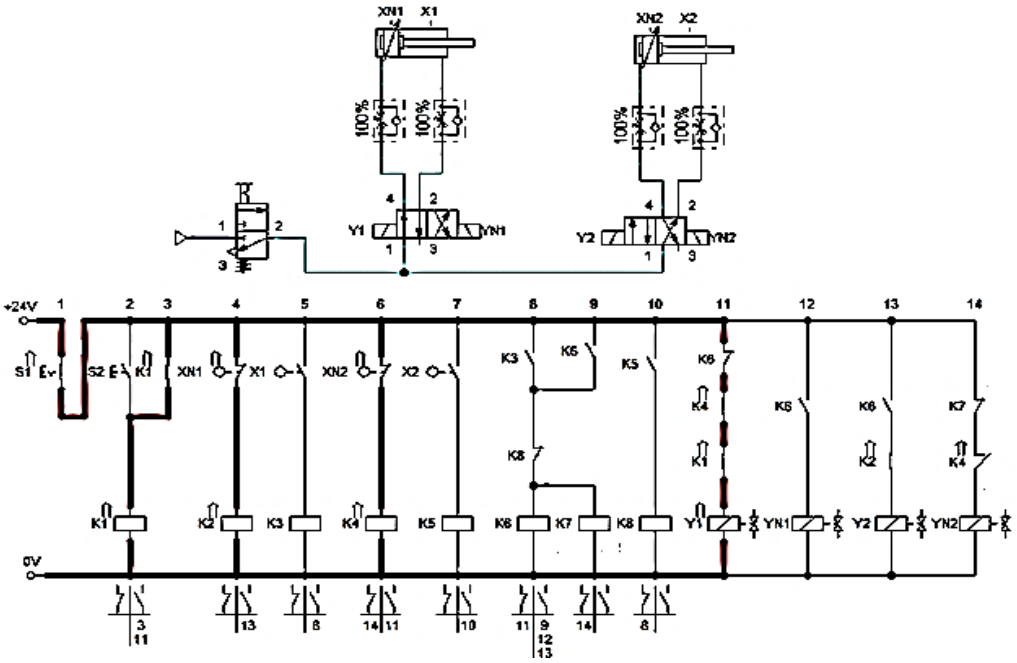


Рис. 8.50. Комбінована схема циклу $\mathbf{1 \rightarrow \overline{1} \rightarrow 2 \rightarrow \overline{2}}$ при вимкненому стані пневматичної схеми і включеному стані електричної схеми

Для якого моменту циклу наведений на рис. 8.51 наведений стоп-кадр циклу $1 \to \overline{1} \to 2 \to \overline{2}$ при включеному стані пневматичної схеми і включеному стані електричної схеми ?

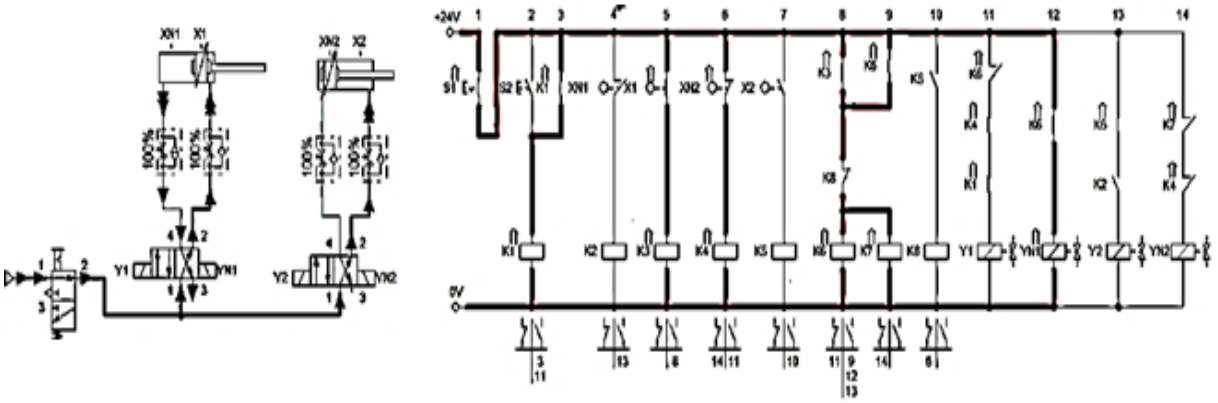


Рис. 8.51. Стоп-кадр циклу $\mathbf{1 \to \overline{1} \to 2 \to \overline{2}}$ при включеному стані пневматичної схеми і включеному стані електричної схеми

На рис. 8.52 наведена функціонально-адекватна пневматична схема (тип П3) циклу $\mathbf{1 \to \overline{1} \to \overline{2} \to 2}$ при включеному компресорі і не натиснутої пневмокнопки.

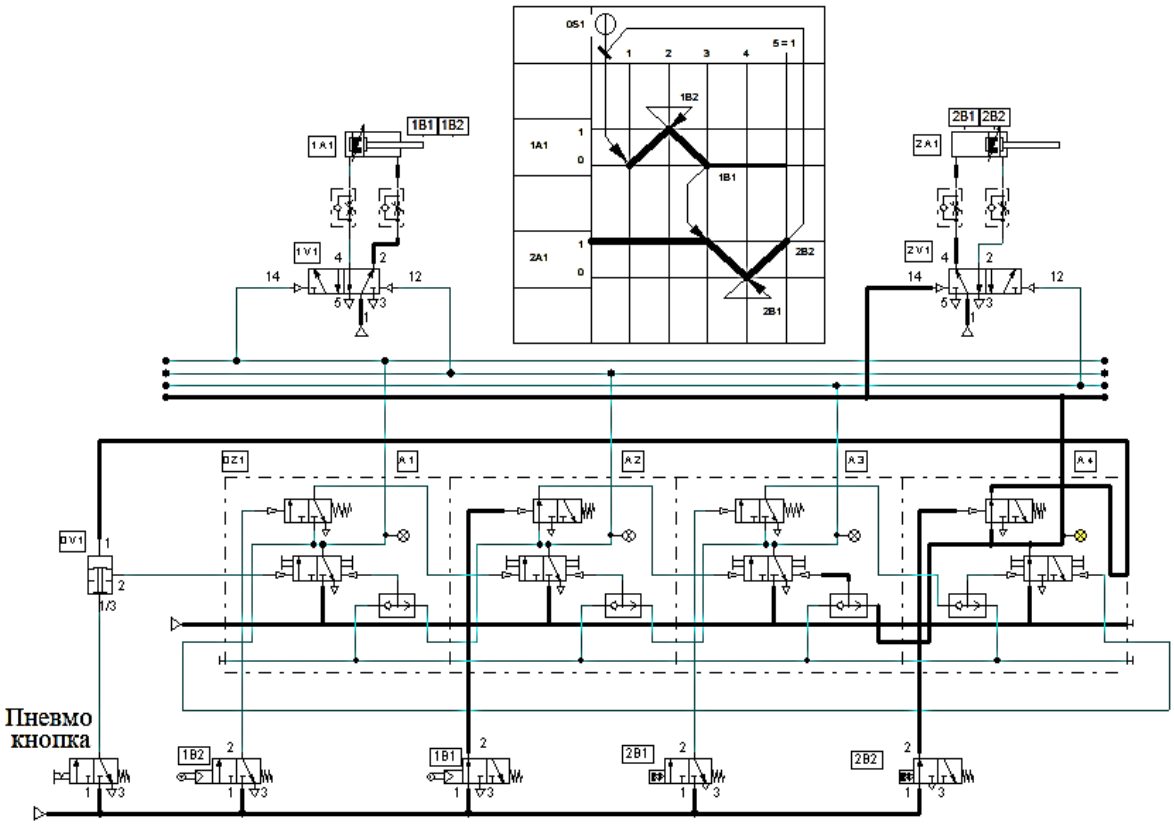


Рис. 8.52. Функціонально-адекватна пневматична схема (тип П3) циклу

$\mathbf{1} \rightarrow \overline{\mathbf{1}} \rightarrow \overline{\mathbf{2}} \rightarrow \mathbf{2}$ при включеному компресорі і не натиснутої пневмокнопки

8.12. **БІстабільне керування циклом 1 – N1– 2 – N2 на типових пневматичних функціональних модулях мехатроніки**

В залежності від умов експлуатації механіко-технологічних систем і машин проект для одно режимного циклу **1-N1-2-N2** можна реалізувати без контролера на елементах і модулях електромагнітних і пневматичних або тільки на елементах і модулях пневматичних (для вибухонебезпечних виробництв).

Розглянемо приклад створення проекту для типового однорежимного циклу **1-N1-2-N2** на пневматичних елементах і модулях з використанням програмне середовище FluidSim-**P** . На рис. 8.53 наведена пневматична схема (тип П3) циклу **1 – N1– 2 – N2** з БІстабільним керуванням та двома виконавчими механізмами.

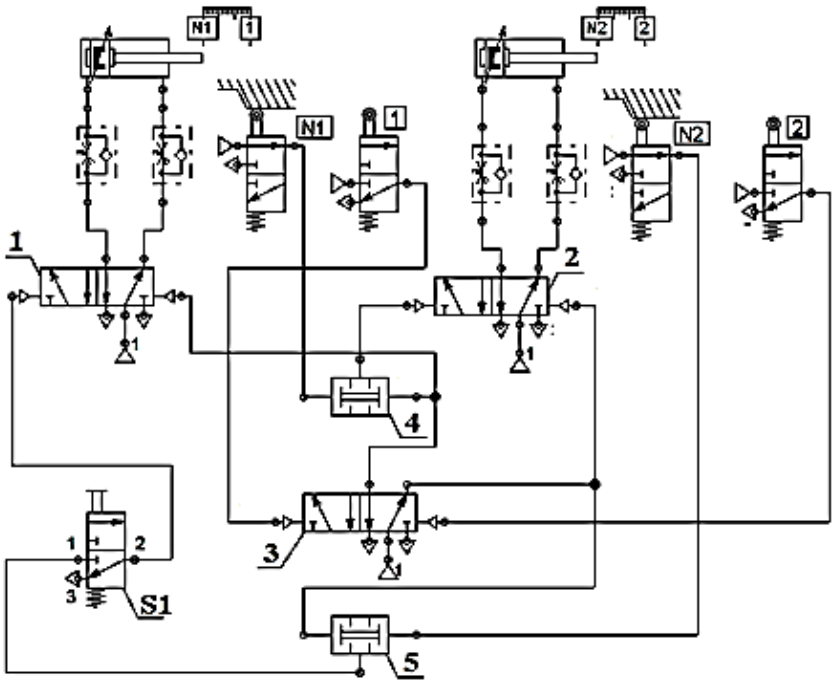


Рис. 8.53. Пневматична схема (тип П3) циклу **1 – N1– 2 – N2** з БІстабільним керуванням та двома виконавчими механізмами: **S1** – кнопка «Start»; **1, 2** і **3** – пневморозподільники типу 5/2; **4** і **5** – логічні елементи «**AND**»

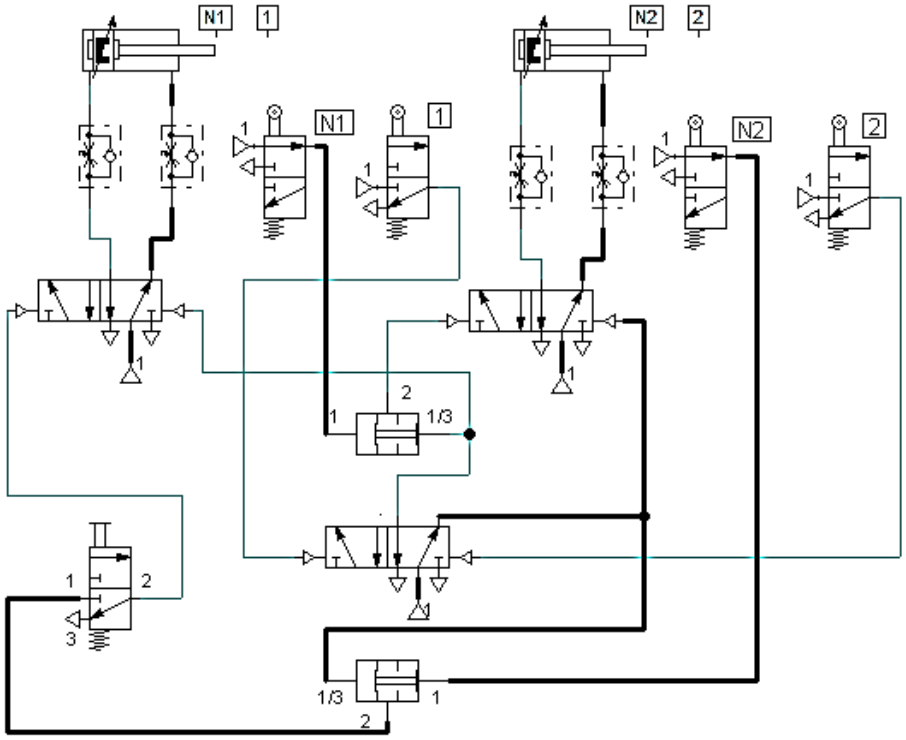


Рис. 8.54. Початковий стан схеми на рис. 8.53 після включення компресора і не натисканній пневмокнопки «Start» S1

З рис. 8.53 випливає вихідне втягнуте положення штоків поршнів двох виконавчих механізмів і відкритий/закритий стан їх кінцевих вимикачів. Логічний елемент 5 (рис. 8.53) має на виході стан «1» тому, що на його двох входах присутній також логічний стан «1» (*жирними лініями виділені канали під тиском стислого повітря* на рис. 8.54).

Після одноразового натискання МОНОстабільної кнопки S1 схема на рис. 8.54 приймає стан, наведений на рис. 8.55 (крок 1) – початок руху поршня першого пневмоцилінду. Всі наступні крокі делегування обов'язків елементів мехатроніки в термінах об'єктно-орієнтованого проектування наведені на рис. 8.56 … рис. 8.60.

На кроці 2 (рис. 8.56) спрацьовує (відкривається) кінцевий вимикач 1 , спрацьовує пневморозподільник 3, елементі пам'яті і логічний елементи «AND» 4 на правому вході переходить в стан «1», а пневморозподільник 1 і перший виконавчий механізм повертаються у попередній стан (крок 3) – рис.8.57.

Тому відкривається вихід логічного елементу 4 (стан «1») , спрацьовує пневморозподільник 2 і шток поршню другого пневмо-циліндру рухається вперед – крок 4 (рис. 8.58). В крайньому висунутому

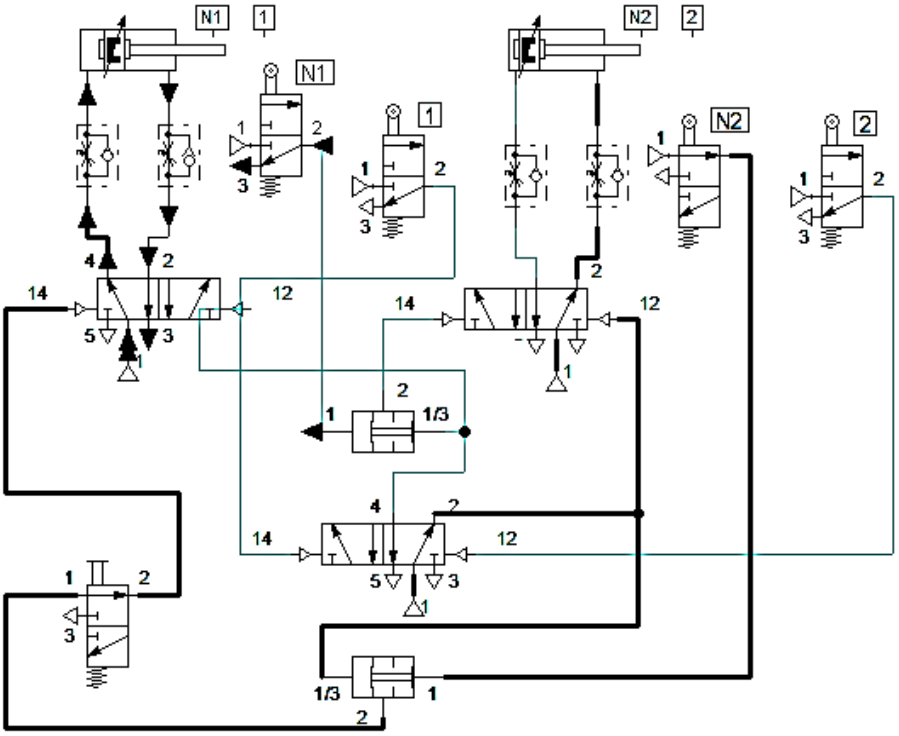


Рис. 8.55. Пневматична схема циклу **1 – N1– 2 – N2 (**крок 1**) –** натиснута пнемокнопка і початок руху поршня першого пневмоцилінду

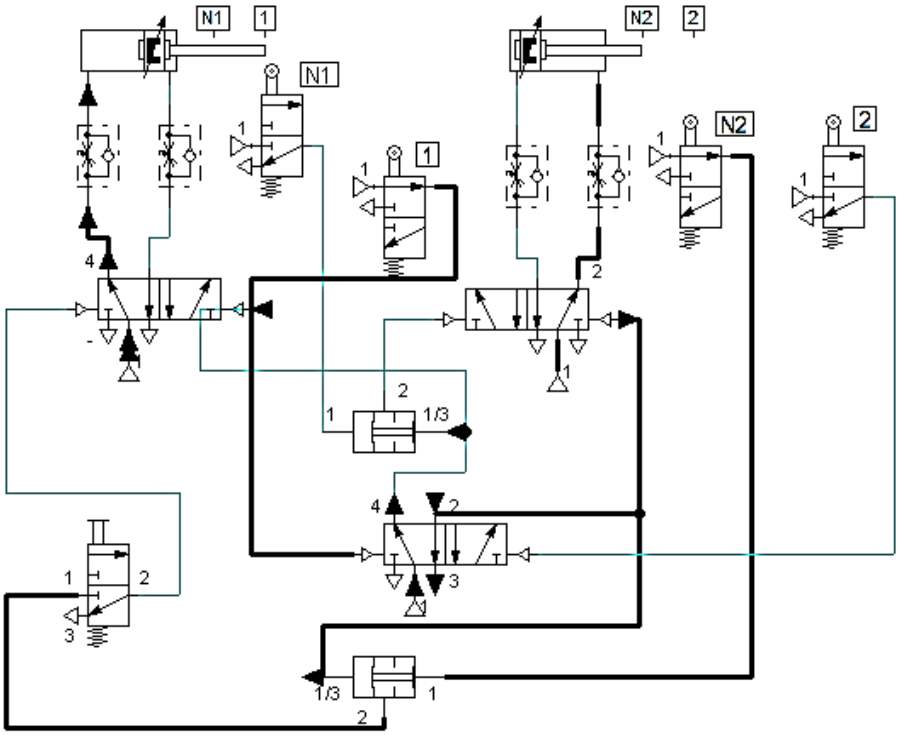


Рис. 8.56. Пневматична схема циклу **1 – N1– 2 – N2 (**крок 2**) –** шток першого пневмоциліндру вмикає пневмокінцевик 1

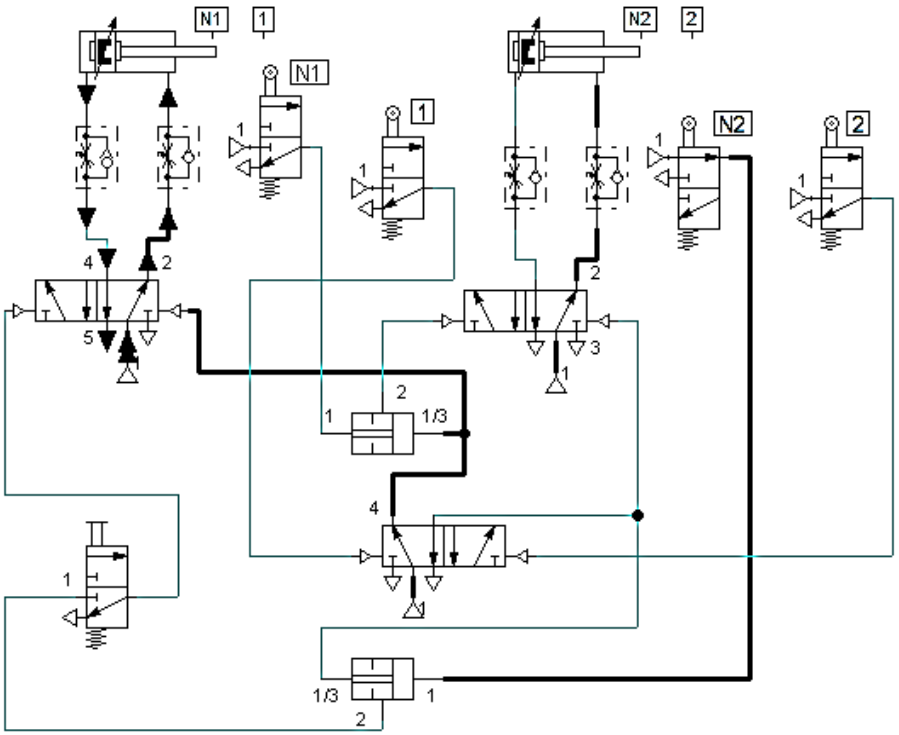


Рис. 8.57. Пневматична схема циклу **1 – N1– 2 – N2 (**крок 3**)** – поршень першого пневмоциліндру повертається назад

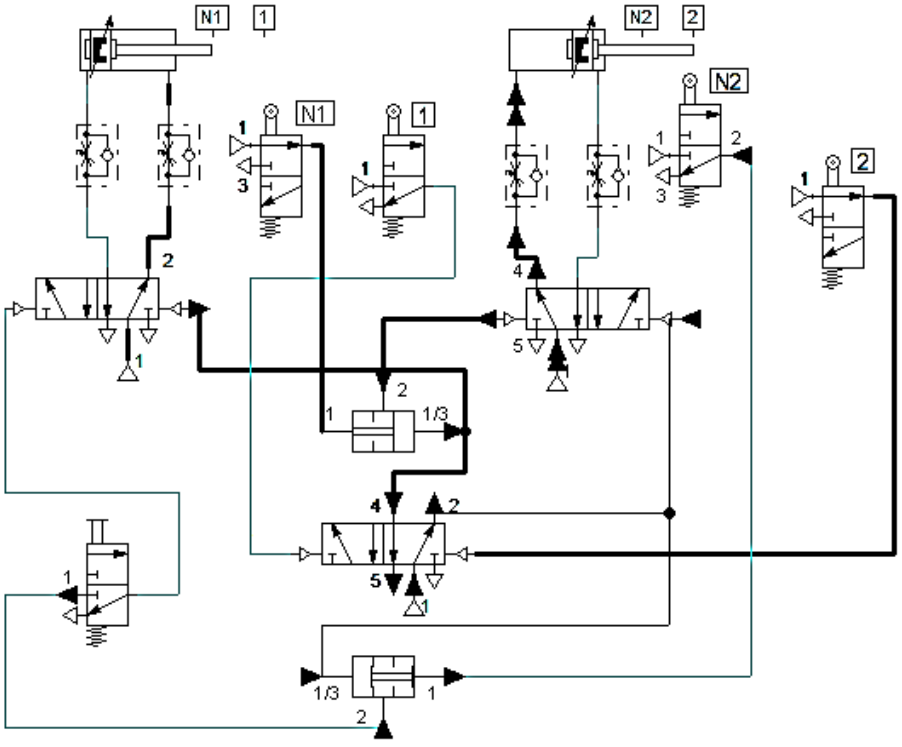


Рис. 8.58. Пневматична схема циклу **1 – N1– 2 – N2 (**крок 4**)** – шток поршню другого пневмоциліндру висунувся вперед

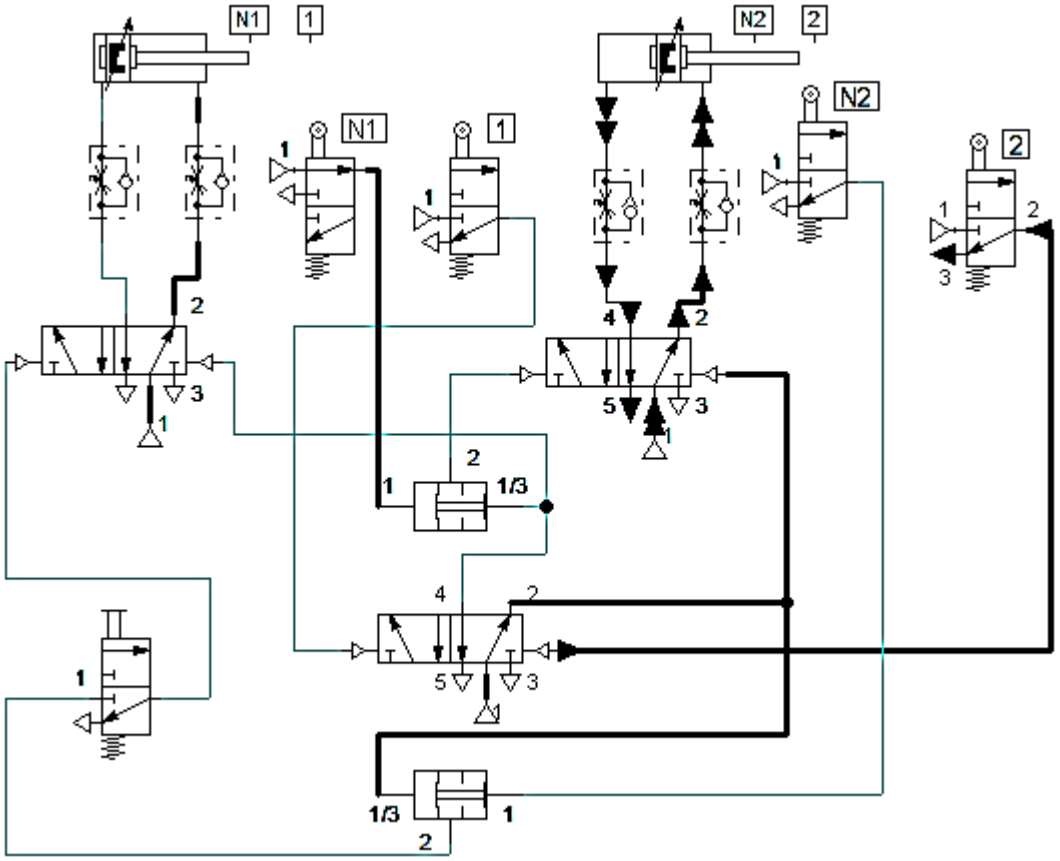


Рис. 8.59. Пневматична схема циклу **1 – N1– 2 – N2 (**крок 5**) –**повернення поршня другого пневмоциліндру назад

положенні штоку спрацьовує (відкривається) кінцевий вимикач 2 і схема переходить в стан на рис. 8.59 – крок 5. Пневморозподільники 2 і 3 повертаються у вихідний стан і поршень другого пневмоциліндру починає рух назад для завершення циклу (рис. 8.60 – крок 6). Схема повертається у початковий стан, наведений на рис. 8.54.

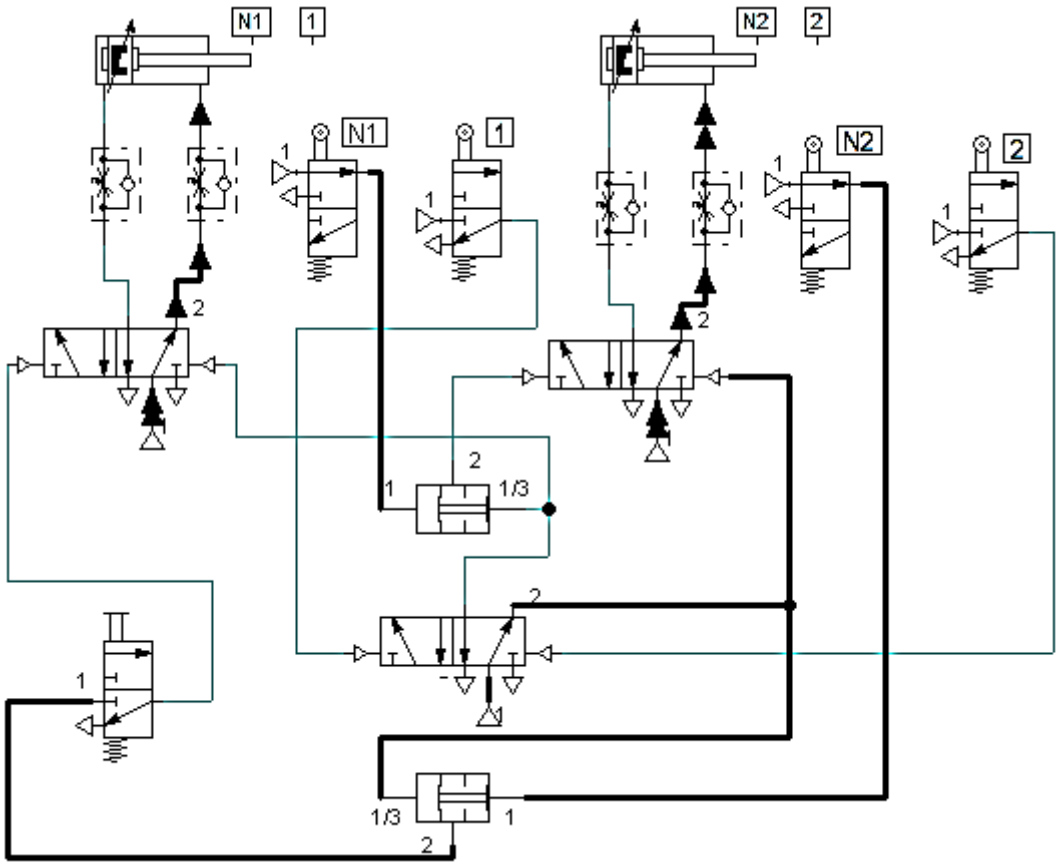


Рис. 8.60. Пневматична схема циклу **1 – N1 – 2 – N2 (**крок 6**) –** повернення виконавчих механізмів в початковий стан (закінчення циклу)

Наведені на рис. 8.56 … рис. 8.60 схеми крокових змін станів функціональних модулів при реалізації циклу **1 – N1 – 2 – N2** дозволяє перевіряти працездатність роботи обраних елементів і виконувати імітації їх роботи в режимі реального часу з обраною швидкістю переміщення ведених ланок механізму, а при необхідності – із зупинкою в крайніх положеннях ведених ланок механізмів технологічних машин, які можуть бути кінематичне з'єднані зі штоками пневмоциліндрів.

При наявності наведених елементів і модулів доцільно постановка і виконання лабораторної роботи **БІстабільне керування циклом 1–N1–2––N2** на типових пневматичних функціональних модулях мехатроніки.

Питання для самоконтролю по розділу 8

1. Яке призначення програмного середовища **FluidSim** ?

2. Чим відрізняється БІстабільне керування від МОНО-стабільного керування ?

3. Чим відрізняється пряме і непряме керування циклами ?

4. Що таке комбінована схема мехатроніки і яке позначення мають такі схеми по ДСТУ ?

5. Які механізми машин легкої промисловості можуть заміняти цикли 1- N1 мехатроніки ?

6. Як позначаються контакти реле в ланцюгах схем типу Е3 при роботі з **FluidSim** ?

7. Яке призначення дроселів і фітингів та як змінити продуктивність дроселю програмним засобом у **FluidSim** ?

8.Як прочитати рівняння (8.21…8.26) ?

9.Запишить команди для циклу $1 \to \overline{1} \to 2 \to \overline{2}$ при МОНО-стабільному керуванні ?

10. Яке призначення елементів «AND» 4 і 5 на рис. 8.53 ?

11. Чим відрізняється робота схеми на рис. 8.61 від роботи схеми на рис. 8.49 ?

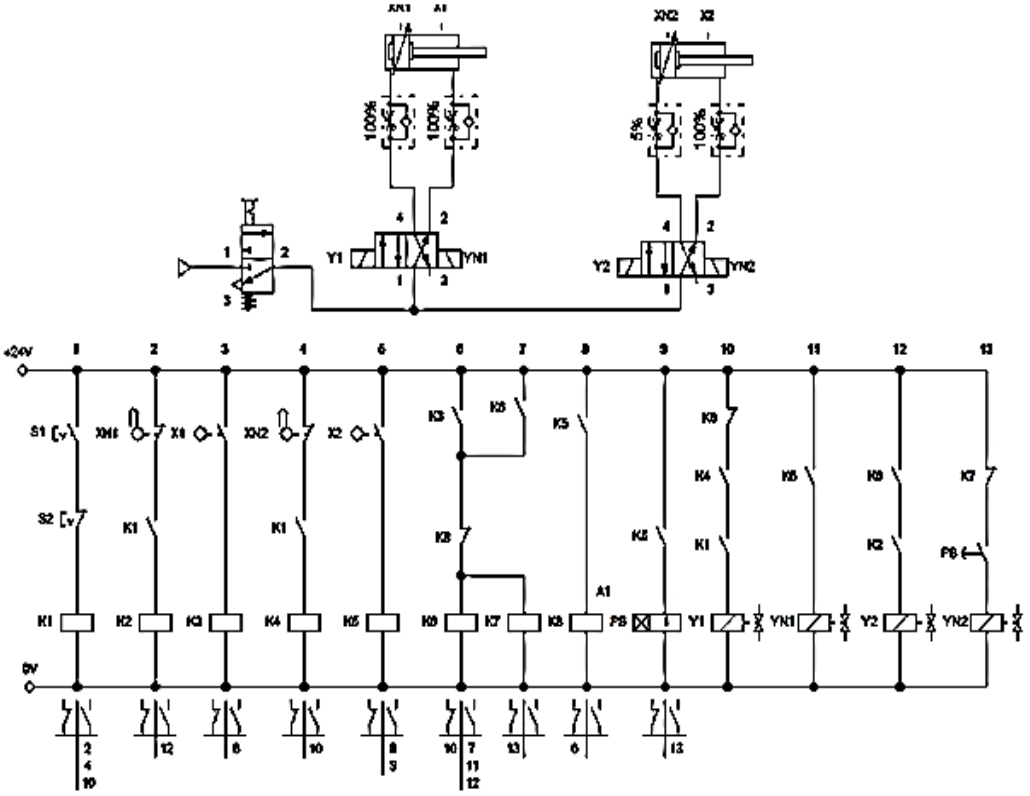


Рис.8.61.

9. БІСТАБІЛЬНЕ І МОНОСТАБІЛЬНЕ КЕРУВАННЯ ЦИКЛАМІ З ПРОГРАМОВАНИМ ЛОГІЧНИМ КОНТРОЛЕРОМ ТЕХНОЛОГІЧНИХ МАШИНАХ ЛЕГКОЇ ПРОМИСЛОВОСТІ

Виконання наведених в розділі проектів здійснено з використанням лабораторного стенду (рис. 9.1), в тому числі програмованого логічного контролера, типових програмованих мехатронних модулів (рис. 9.2) і узагальненого Allocation List (табл. 9.1).

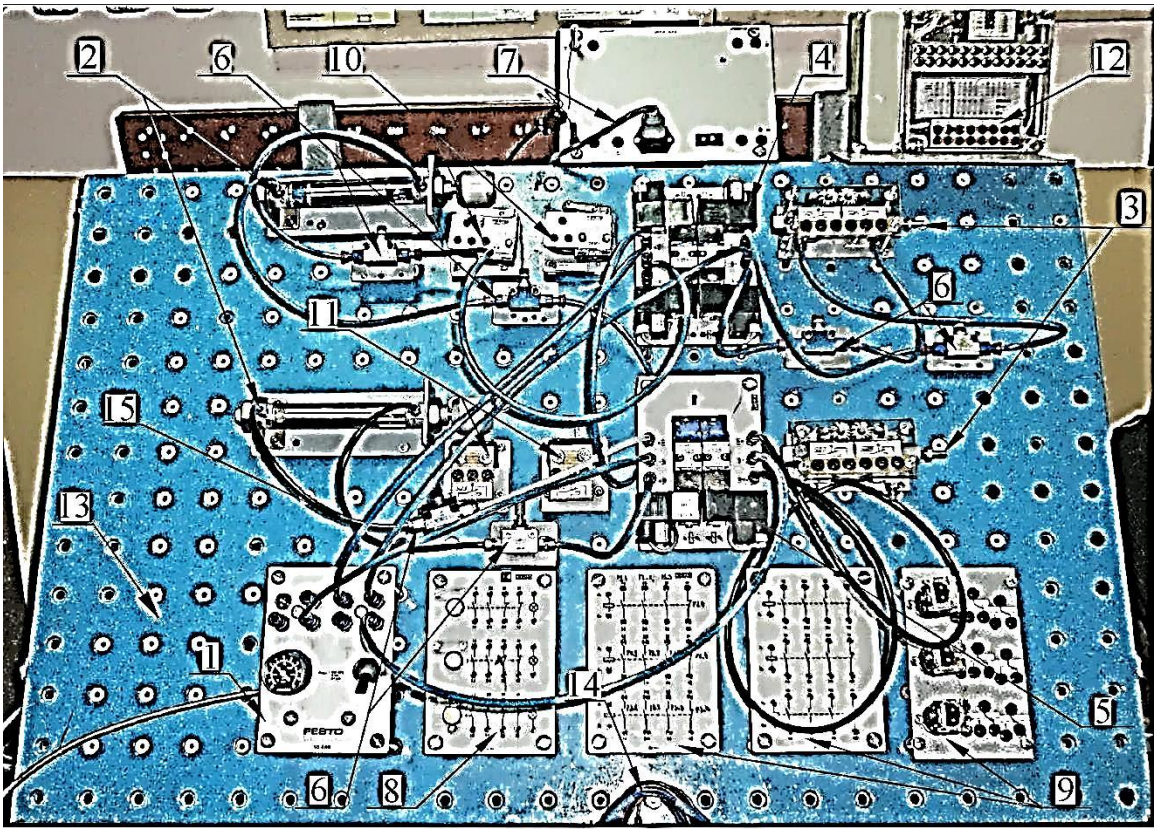


Рис. 9.1. Фото лабораторного стенду:

1 – колектор пневматичний з манометром;

2 – два пневмоциліндри (L = 100 мм, d = 25 мм);

3 – два пневмоциліндри (L = 10 мм, d = 10 мм) з герметичними безконтактними кінцевими вимикачами (герконами);

4 – два пневморозподільники типу 5/2 (п’ятилінійні двохпозиційні) з *БІстабільним* електричним керуванням;

5 – два пневморозподільники типу 5/2 (п’ятилінійні двохпозиційні) з *МОНОстабільним* електричним керуванням;

6 – дроселі пневматичні зі зворотними клапанами;

7 – блок електричного живлення +24 В;
8 – блок кнопок;
9 – три блоки електромагнітних реле (3 × 3 = 9 реле);
10 і 11 – датчики кінцевого положення пневматичних циліндрів;
12 – програмований логічний контролер Festo FC34;
13 – перфорована плита;
14 – набір дротів для складання електричних схем (тип Е3);
15 – трубки для складання пневматичних схем (тип П3)

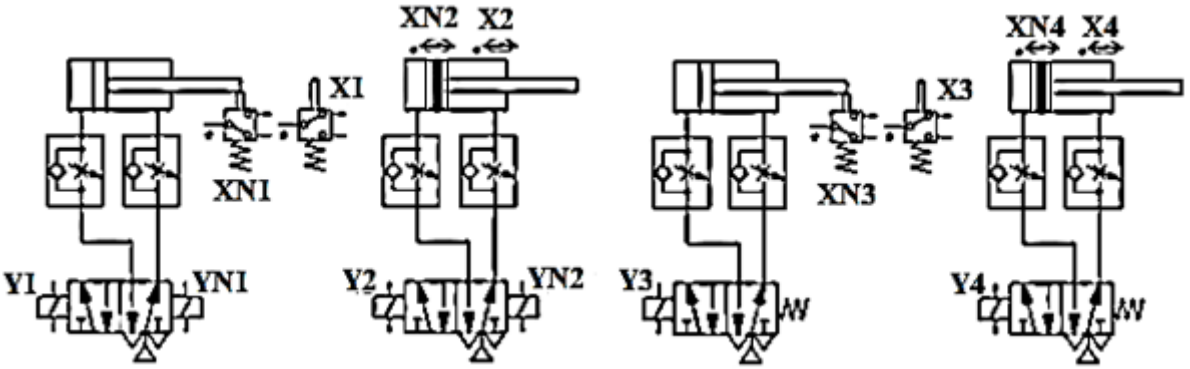


Рис. 9.2. Приклади типових програмованих мехатронних модулів лабораторного стенду

При використанні програмуемого логічного контролера (далі ПЛК, PLC або просто контролер) змінні типу **Boolen** (Булеві змінні) прописуються в Allocation List . Змінна типу **Boolen** може приймати два стану: **true** (істина або логічна одиниця), **false** (хибний або логічний нуль).

ALLOCATION LIST – («СПИСОК РОЗПОДІЛУ») це таблиця, у який прописуються фізичні елементи і програмні елементи проекту. *Фізичні елементи* прописуються як абсолютні імена операндів і регістрів портів виходів (**Output**) і входів (**Input**) контролера, до яких підключені електромагніти пневмоклапанів (**Yi, YNi**), датчики (**Xi, XNi**), кнопки (**Si),** котушки реле (**Ki**). *Програмні елементи*, які зберігаються в пам'яті контролера і не підключаються до портів контролера прописуються наступними операндами: (**Fi**) «прапор» (скорочення від **F**lag, або поширена назва «флаг»), таймер(**Tn**), лічильник (**Cn**), преселектор лічильників (**CPn**), слово лічильників (**CWn**).

Елементи пам’яті (**EPn**) це також не фізичні елементи, а програмні елементи у вигляді двійкової змінної, що має значення «1» - елемент пам’яті включений або значення «0» - елемент пам’яті вимкнений. В Allocation List та в програмі елементи пам’яті (**EPn**) прописуються прапором з ім’ям **Flag**, а тому операндами (**Fi**). Для кнопки Start XST використовується елемент пам’яті (операнд **F0.0**).

Для кожного проекту розділу 9 з таблиці 9.1 обираються потрібні операнди і тому далі наведена вказівка про потребу складання Allocation List з урахуванням таблиці 9.1.

Таблиця 9.1 Allocation List

Operand	Symbol	Comment
O0.0	**Y1**	Прямий хід поршню 1-го циліндра
O0.1	**YN1**	Зворотній хід поршню 1-го циліндра
O0.2	**Y2**	Прямий хід поршню 2-го циліндра
O0.3	**YN2**	Зворотній хід поршню 2-го циліндра
O0.4	**Y3**	Прямий хід поршню 3-го циліндра
O0.5	**Y4**	Прямий хід поршню циліндра 4
I0.0	**XN1**	Кінцевий вимикач циліндру 1 (втягнутий стан поршню)
I0.1	**X1**	Кінцевий вимикач циліндру 1 (витягнутий стан поршню)
I0.2	**XN2**	Кінцевий вимикач циліндру 2 (втягнутий стан поршню)
I0.3	**X2**	Кінцевий вимикач циліндру 2 (витягнутий стан поршню)
I0.4	**XN3**	Кінцевий вимикач циліндру 3 (втягнутий стан поршню)
I0.5	**X3**	Кінцевий вимикач циліндру 3 (витягнутий стан поршню)
I0.6	**XN4**	Кінцевий вимикач циліндру 4 (втягнутий стан поршню)
I0.7	**X4**	Кінцевий вимикач циліндру 4 (витягнутий стан поршню)
I1.0	**S1**	Кнопка 1 без фіксатора
I1.1	**S2**	Кнопка 2 без фіксатора

I1.2	**S3**	Кнопка 3 з фіксатором
F0.0	**XST**	Елемент пам`яті
F0.1	**EP1**	Елемент пам`яті 1
F0.2	**EP2**	Елемент пам`яті 2
F0.3	**EP3**	Елемент пам`яті 3
T0	**Tn**	Таймер 0
T1	**Tn**	Таймер 1
C0	**Cn**	Лічильник 0
CP0	**CPn**	Преселектор лічильника 0
CW0	**CWn**	Слово лічильника 0

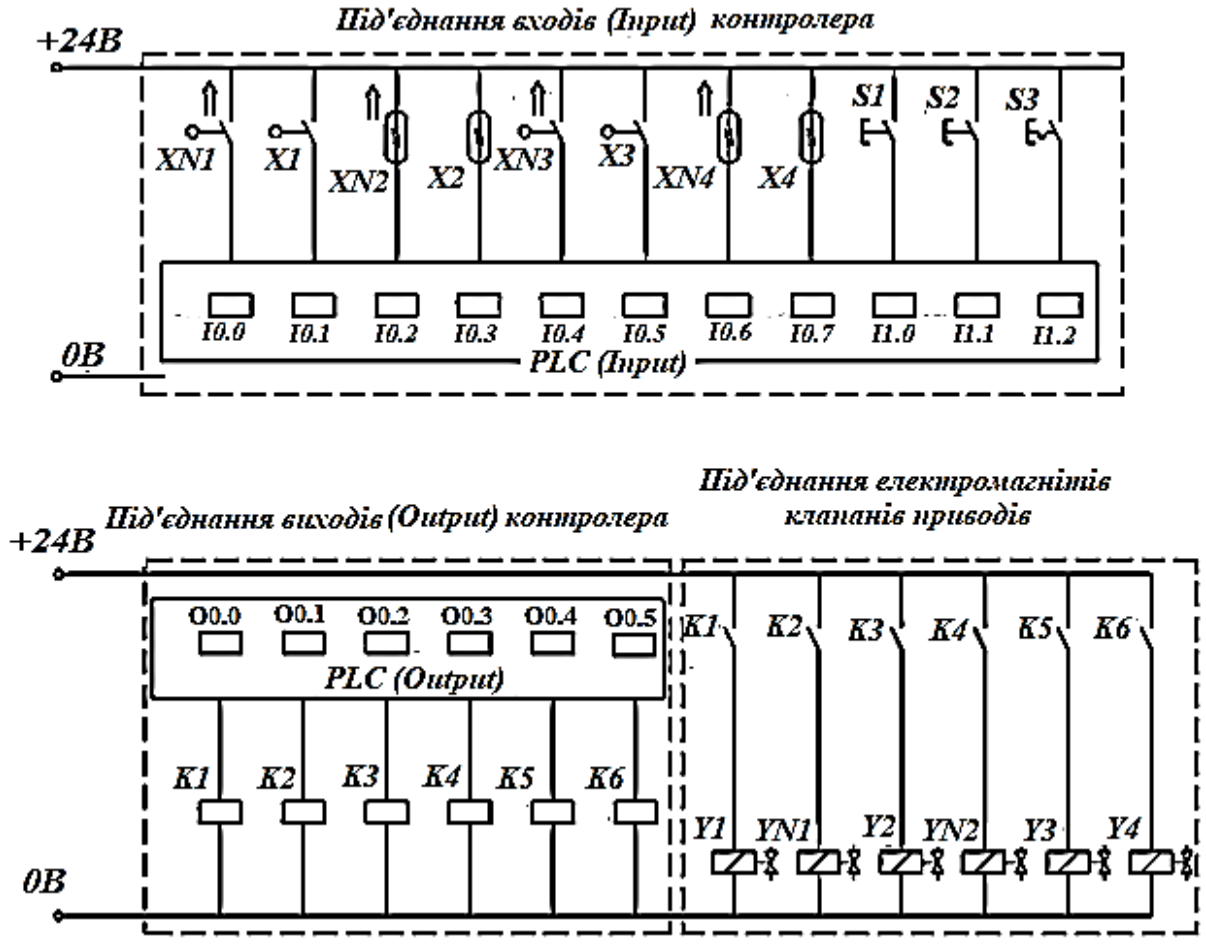


Рис. 9.3. Приклад підключення датчиків, реле, електромагнітів пневматичних розподільників до шини входів (Input), шини виходів (Output) та джерела живлення

При розробці проектів з БІстабільним керування або МОНОстабільним керуванням передбачено розробка програм мехатронних систем згідно з технічним завданням на проектування. Після складання і уточнення технологічного (функціонального) графу виконується складання комбінованої схеми з контролером. На рис. 9.3 наведений приклад підключення датчиків положення (кінцевих вимикачів) *XI…X3, XN1…XN3*, кнопок *S1, S2* і *S3*, реле *K1…K6*, електромагнітів *YI…Y3, YN1…YN3* пневматичних розподільників до шини входів (Input), шини виходів (Output) та джерела живлення 24 В контролера.

Таким чином, в схемах мехатроніки без контролера такі елементи як таймери і лічильники є фізичними елементами у вигляді реле часу і електромеханічного або електронного лічильника. А в схемах мехатроніки з контролером, як підкреслено, таймери і лічильники є програмними елементами, які в Allocation List (табл.9.1) прописані операндами **Tn, Cn, CPn** і **CWn.** Всі інші фізичні елементи мехатронних модулів прописані в в Allocation List і зображені у комбінованих схемах з контролером.

БІстабільне керування це двостороннє керування виконавчим механізмом, наприклад, за допомогою БІстабільного пневматичного розподільника (пневматичного розподільника з двома електромагнітами)

МОНОстабільне керування це одностороннє керування виконавчим механізмом, наприклад, за допомогою МОНОстабільного пневматичного розподільника(пневматичного розподільника з одним електромагнітом)

9.1. **Розробка проектів з контролерами для виконання циклу $\mathbf{1 \rightarrow \overline{1}}$ з БІстабільним і МОНОстабільним керуванням виконавчим механізмом та двома кінцевими вимикачами**

Для створення та компіляції програми, написаної на мові **ST** «**S**tatement **L**ist» у середовищі FST4.21 для контролера Festo треба діяти за наступними кроками:

- відкрити новий *New Project* і надати ім’я проекту, латиницею та цифрами (не більше 8 символів);
- обрати тип контролера (FEC Compact);

- скласти *Allocation List* ;
- Ctrl+N (вибрати мову програмування) ;
- написати програму керування циклом;
- перенести проект на виданій викладачем флеш-пам'яті для перенесення та завантаження цього проекту на PLC;
- виконати компіляцію і перевірку програми на помилки (*Errors*), а при наявності помилок їх треба виправити;
- завантажити проект на PLC (Download Project);
- запустити проект;
- зупинити проект;
- вивантажити проект з PLC.

Фрагмент головного меню і фрагмент Allocation List наведені на рис.9.4, де цифрами 1…6 позначені опції для виконання наступних дій :

1 – **Compile Active Module** (компіляція контролера);

2 – **Make Project** (компіляція програми);

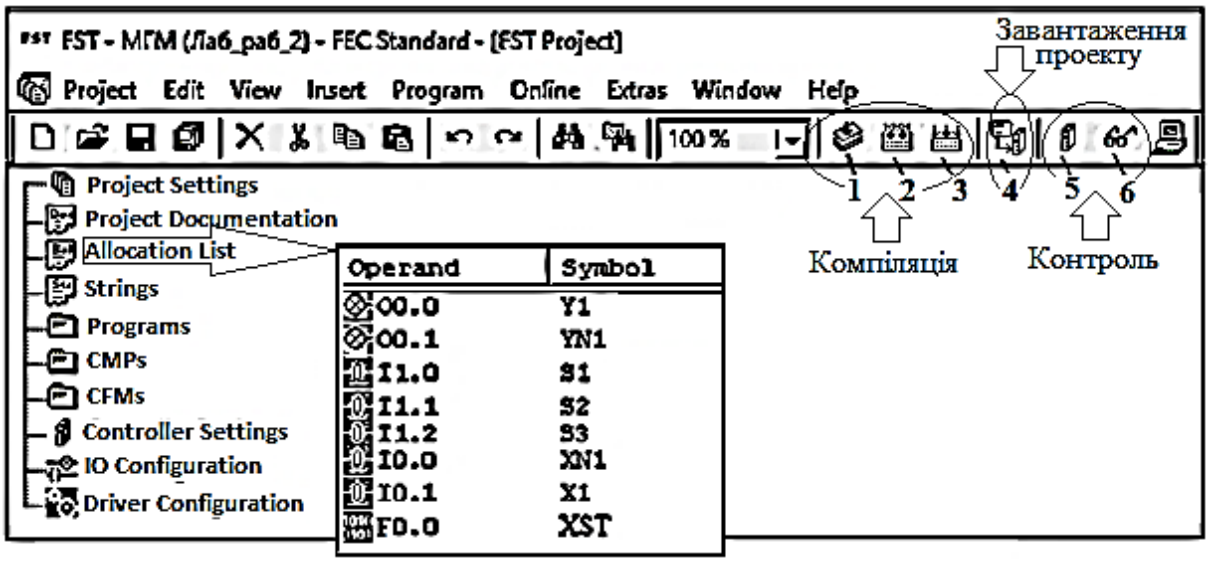
3 – **Build Project** (підготовка проекту до завантаження у PLC);

4 – **Download Project** (завантаження проекту);

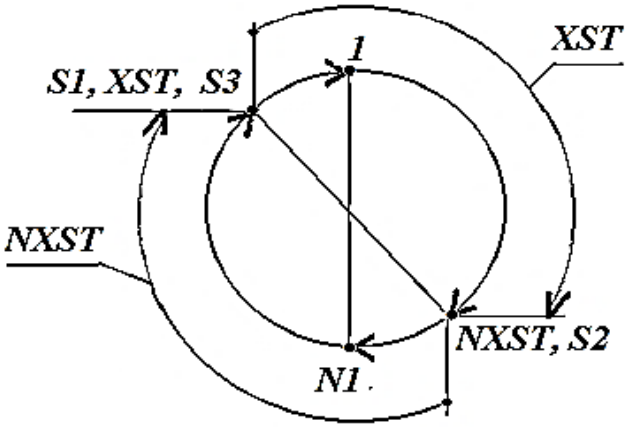
5 – **Control Panel** (контроль стану запуску-зупинки-вивантаження проекту);

6 – **Online Display** (візуалізація значень змінних PLC).

У технологічних машинах легкої промисловості широко застосовуються типові кривошипно-повзунні механізми, кривошипно-коромислові, двокоромислові та двокривошипні механізми, які є виконавчими цикловими важільними механізмами з жорсткою системою керування типу **РВ** (типу «**Р**озподільний **В**ал»). Обов'язковими ланками таких механізмів є ведуча ланка кривошип або ексцентрик і передаточна ланка – шатун, які утворюють кінематичні пари між собою, головним валом і з веденою ланкою (наприклад, з повзуном-голководом або з коромислом механізму ниткопритягувача човникових швейних машин).



а



б

Рис. 9.4. Фрагмент головного меню і фрагмент Allocation List (а) та функціональний граф (б) для циклу $1-\bar{1}$

Мехатронними функціонально-адекватними механізмами є комп'ютерно-інтегровані циклові механізми машин. Такі механізми мають спрощену кінематику і їх ведені ланки з робочим інструментом кінематичне не з'єднані з головним валом, а мають інформаційні (дискретно-електричні) зв'язки з одного боку з головним валом машини, а з іншого боку з портами входів і портами виходів вбудованого в технологічну машину PLC.

В комп'ютерно-інтегрованих технологічних машинах легкої промисловості цикл $1 \to \bar{1}$ можливо реалізовувати без традиційних кривошипів і шатунів циклових механізмів, а з допомогою логічних команд керування при БІстабільному або МОНОстабільному керуванні.

Мета проекту: отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $1 \to \bar{1}$ з БІстабільним керуванням виконавчим механізмом та двома кінцевими вимикачами.

Технічне завдання. Автоматизувати процес роботи комп'ютерно-інтегрованого циклового механізму машини. При натисненні кнопки *«Start» S1* без фіксатора у натиснутому стані – рис. 9.5 і табл. 9.1) і при умові перебування веденої ланки в вихідному положенні і натиснутій кнопки *«Пуск» S3* (кнопка *S3* з фіксатором у натиснутому стані) повинен розпочатися цикл роботи системи. Після переміщення веденої ланки в робоче положення і натиснутій кнопки *«Пуск» S3* ведена ланка повертається в початкове положення, *система допрацьовує останній робочий хід і зупиняється* до наступного натиснення кнопки S1. При натисненні кнопки *«Stop» S2* (кнопка *S2* без фіксатора у натиснутому стані – рис. 9.4 і табл. 9.1) – цикл зупиняється і робота системи не може бути продовжена при повторному натисканні і відпусканні цієї кнопки. У випадку відтиснення кнопки *«Пуск» S3* – *система зупиняється*. При повторному її натиснення система продовжує працювати.

Вибираємо наступне обладнання:

програмований логічний контролер (PLC) *Festo FC34*;

пневматичний циліндр двосторонньої дії;

БІстабільний пневматичний розподільник 5/2 з електромагнітним керуванням (електромагніти *Y1* та *YN1*);

два пневматичні дроселі зі зворотними клапанами;

два кінцевих вимикача (наприклад, для контролю крайніх положень голководу кривошипно-повзунного механізму голки швейних машин) з електричними контактами *XN1* та *X1*;

кнопку «Start» *S1*, кнопку «Stop» *S2* і кнопку «Пуск» *S3*;

електромагнітні реле *K1* і *K2*;

джерело живлення постійним струмом (***DC*** – ***D****irect* ***C****urrent*) 24 В.

З цих елементів потрібно скласти комбіновану схему, яка наведена на рис. 9.5.

Позначимо кінцевий вимикач, який відповідає за витягнуте положення штока $\boldsymbol{X1}$, а кінцевий вимикач, що відповідає за втягнуте положення $\boldsymbol{XN1}$. При натисненні штока поршню на кінцевий вимикач $X1$ замикається його електричний контакт. В такому випадку приймається, що логічна змінна $X1 = 1$. При відтиснені кінцевого вимикача контакт розмикається, а логічна змінна приймає значення $X1 = 0$. Електромагніт $\boldsymbol{Y1}$ відповідає за прямий рух поршню пневматичного циліндру, а електромагніт, $\boldsymbol{YN1}$ відповідає за зворотній рух. Таким чином, при подачі сигналу на електромагніт $\boldsymbol{Y1}$ його логічна змінна $Y1 = 1$, при знятті сигналу $Y1 = 0$. Аналогічно для електромагніту $\boldsymbol{YN1}$.

На функціональному графі (рис. 9.4,б) крапками позначені вершини графу, а стрілками – ребра графу. При складанні команд програм потрібно аналізувати операнди на початку і на кінці ребер (стрілок) графу.

Схема з контролером для виконання циклу згідно завдання, який відтворює роботу кривошипно-повзунного механізму без кривошипу і без шатуна наведена на рис. 9.5.

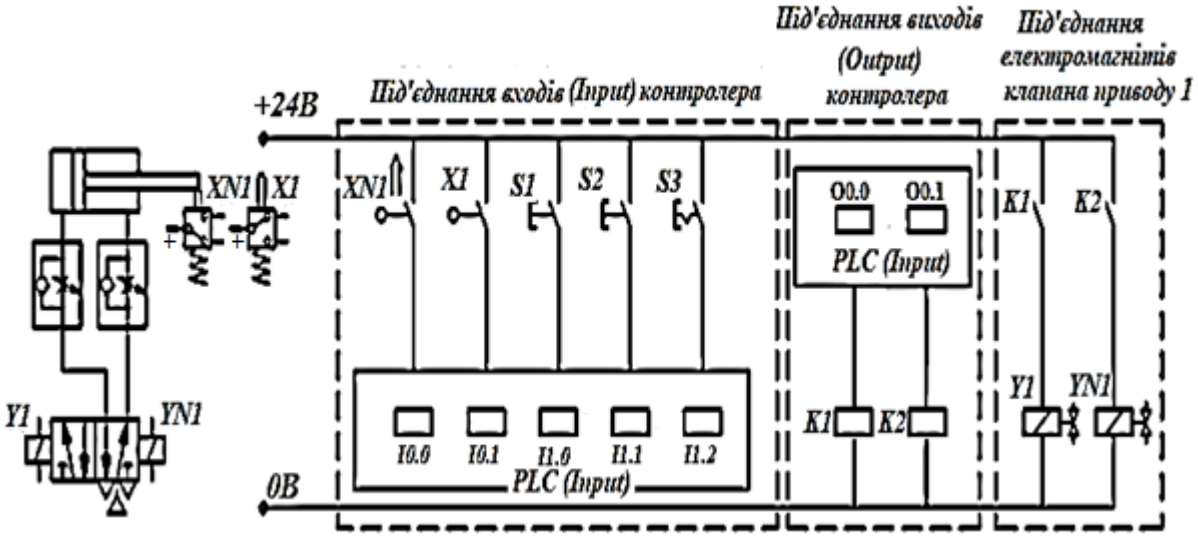


Рис. 9.5. Комбінована схема (тип С3) з контролером для виконання циклу $1 \to \bar{1}$ з БІстабільним керуванням виконавчим механізмом та двома кінцевими вимикачами

Складання логічних команд керування: домовимось, що такий цикл читати треба наступним чином: «*один – не один*» і позначати $1 \to \bar{1}$, або

$1 \rightarrow N1$. Для початку роботи при натисненні кнопки «Start» **S1** повинен увімкнутись елемент пам'яті *XST*, який буде відповідати за Start роботи системи:

$$\boldsymbol{XST \Leftarrow S1}, \tag{9.1}$$

де позначка «$\Leftarrow$ » має зміст причинно-наслідкового зв'язку змінних, а саме істинності операнду (дорівнюється «1»). Вираз (9.1) можна прочитати наступним чином: «*Якщо нетиснена кнопка «Start»* $\boldsymbol{S1}$*, тоді включений елемент пам'яті* $\boldsymbol{XST}$», або «*При натисканні кнопки «Start»* $\boldsymbol{S1}$ *потрібно включити елемент пам'яті* $\boldsymbol{XST}$», або «*При натисненні кнопки «Start»* $\boldsymbol{S1}$ *елемент пам'яті* $\boldsymbol{XST}$ включений».

Ведена ланка (шток поршня), перебуваючи у втягнутому положенні повинна почати робочий хід при наявності сигналу елементу пам'яті $\boldsymbol{XST}$ згідно з виразом (9.1) та першого натискання і фіксації у включеному стані кнопки «Пуск» $\boldsymbol{S3}$ для початку циклу і включення електромагніту $\boldsymbol{Y1}$ пневматичного розподільника(пряма команда). Тоді рівняння причино-наслідкових зв'язків має наступний вигляд:

$$\boldsymbol{Y1 \Leftarrow XST \cdot XN1 \cdot S3} \tag{9.2}$$

При другому натисканні кнопки «Пуск» $\boldsymbol{S3}$ і замкненому кінцевому вимикачу $\boldsymbol{X1}$ завершується цикл і ведена ланка повернеться в початкове положення. При цьому логічна зворотна команда має наступний вигляд:

$$\boldsymbol{YN1 \Leftarrow X1 \cdot S3} \tag{9.3}$$

Для вимикання системи необхідно натиснути кнопку $\boldsymbol{S2}$ і тоді вимкнеться елемент пам'яті, який відповідає за початок роботи системи:

$$\boldsymbol{NXST \Leftarrow S2} \tag{9.4}$$

У випадку необхідності екстреної зупинки системи потрібно відтиснути кнопку «Пуск» $\boldsymbol{S3}$.

З а у в а ж е н н я: позначкою «'» на початку кожного рядка в програмі тут і надалі позначені коментарі до програми, які ігноруються транслятором.

Розробка програми

```
'Створюємо STEP 0, в якому приводимо систему в
'початковий стан (NOP:= No Operation) при будь-яких
'умовах, а саме вмикаємо електромагніт зворотного
'ходу SET YN1, вимикаємо електромагніти прямого ходу
'RESET Y1, а також вимикаємо елемент пам'яті XST, які
'будуть використані в робочому кроці STEP 1
'Для переходу (JMP TO 1) в робочий крок STEP 1
'необхідно щоб  шток поршню циліндру був втягнутим,
'тобто це необхідна умова (IF XN1) для замкненого
'стану кінцевого вимикача XN1

STEP 0
IF                  NOP
THEN  SET           YN1
      RESET         Y1
      RESET         XST
IF                  XN1
THEN  JMP TO 1

'В робочому кроці STEP 1 записуємо раніше складені
'команди (9.1)…(9.4). Важливо пам'ятати, що при
'керуванні приводами з БІстабільним керуванням
'необхідно в одній команді вмикати один сигнал і
'вимикати інший, тобто при керуванні БІстабільним
'пневморозподільником є недопустимим щоб одночасно
'виконувались умови Y1 = 1  та YN1 = 1

STEP 1
IF                  S1
      AND      N    S2
THEN  SET           XST       'рівняння (9.1)

IF                  XST
      AND           XN1
      AND           S3
THEN  SET           Y1        'рівняння (9.2)
      RESET         YN1
```

```
IF                    X1
      AND             S3
THEN  SET             YN1        'рівняння (9.3)
RESET          Y1

IF                    S2
THEN  RESET           XST        'рівняння (9.4)

'Важливо пам'ятати, що при програмуванні останній
'крок програми необхідно завершувати командою на
'переведення системи в крок STEP 1 або в наступний
'крок при багатокрокової структурі програми

IF                    NOP
THEN  JMP TO 1
```

Реалізація програми. Увімкнути компресор для створення тиску в пневмомережі (4 бара = 0.4 МПа) для подачі енергоносія до пневморозподільника. Увімкнути джерело живлення 7 лабораторного стенду (рис.9.1) та перевірити правильність виконання запрограмованого циклу роботи веденої ланки циклового механізму машини.

Розглянемо варіант МОНОстабільного керування циклом $1-\bar{1}$ у відповідності до комбінованої схеми на рис. 9.6, яку також можна розглядати, як мехатронний функціонально-адекватний аналог кривошипно-повзунного механізму, наприклад, механізму голки швейних машин.

Таблиця 9.2 Allocation List

Operand	Symbol	Comment
O0.0	**Y1**	Прямий хід поршню пневмо циліндру
O0.1	**YN1**	Зворотній хід пневмо циліндру
I 0.0	**S1**	Кнопка 1 без фіксатора
I 0.1	**S2**	Кнопка 2 без фіксатора
I 0.2	**XN1**	Кінцевий вимикач циліндру для втягнутого стану поршню
I 0.3	**X1**	Кінцевий вимикач циліндру для витягнутого стану поршню
F 0.0	**EP1**	ELEMENT PAMYATI 1

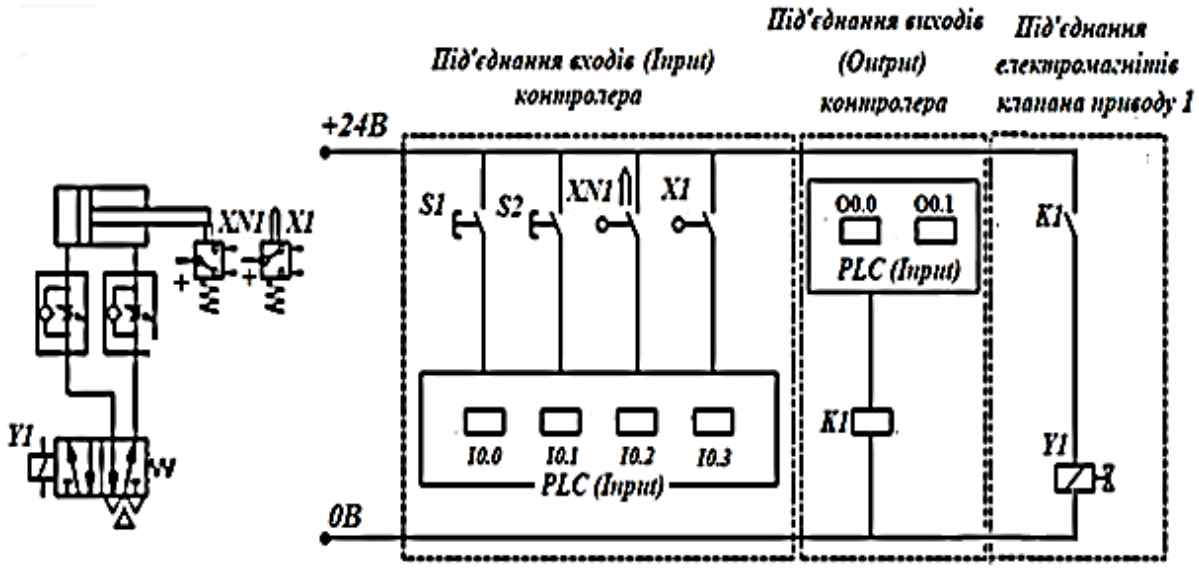


Рис. 9.6. Комбінована схема з контролером для виконання циклу $1 \to \bar{1}$ з МОНОстабільним керуванням виконавчим механізмом та двома кінцевими вимикачами

Розробка програми

```
'Підготовка програми до пуску це приведення системи в
'початкове положення. При приведені системи в
'початкове положення Step 0 перейти до кроку 1

STEP 0
IF                    NOP
 THEN  RESET          Y1
       RESET          EP1          'ELEMENT PAMYATI 1
         SET          YN1
 IF                   XN1
 THEN  RESET          YN1
       JMP TO 1

'При натисненні кнопки S1 увімкнути елемент пам'яті
'EP1, який вказує на початок роботи системи

STEP 1
IF                    S1           'START
 THEN  SET            EP1          'ELEMENT PAMYATI 1
```

```
'При втягнутому положенні штока XN1 і ввімкненому
'елементі пам'яті  EP1, увімкнути Y1 і вимкнути YN1

 IF                    XN1
       AND             EP1          'ELEMENT PAMYATI 1
 THEN  SET             Y1
       RESET           YN1

'При витягнутому положенні штока X1 і включеному
'елементі пам'яті  EP1, вимкнути Y1 і увімкнути YN1

 IF                    X1
       AND             EP1          'ELEMENT PAMYATI 1
 THEN  RESET           Y1
       SET             YN1

'При натисненні кнопки S2 і ненатисненій кнопці S1
'вимкнути EP1

 IF                    S2            'STOP
       AND     N       S1            'START
 THEN  RESET           EP1           'ELEMENT PAMYATI 1

'При відсутності сигналів перейти до кроку 1

 IF                    NOP
 THEN  JMP TO 1
```

Питання і завдання для самостійної роботи:

1.Чим відрізняється комбінована схема на рис. 9.6 від схеми на рис. 9.5 ?

2. Чим відрізняється схеми **типу С3** від схем **типу Е3** ?

3. Порівняти комбіновану схему з контролером для БІстабільного керуванням виконавчим механізмом та двома кінцевими вимикачами (рис. 9.5) з комбінованою схемою без контролера (рис. 8.8 у розділу 8) по

кількості дротів і кількості контактів електромагнітних реле, які застосовані в цих схемах.

Для порівнювального аналізу програми для циклу $1 \to \overline{1}$ з контролером **Festo FC34** розглянемо програму для циклу $1 \to \overline{1}$ з контролером **Arduino UNO**. Роботу двох кінцевих вимикачів імітують включення/вимкнення світлодіодів **led1 і led2** при одноразовому і дворазовому натисканні кнопки **button. led1:=XN1** (шток поршню виконавчого механізму втягнутий) і **led2:=X1** (шток поршню виконавчого механізму висунутий). Одноразовому натисканні **button** відповідає умова **if (digitalRead(button) == HIGH),** а дворазовому натисканні **button** відповідає умова **if (digitalRead(button) == HIGH && state == true).** При одноразовому натисканні **button** відбувається цикл $\mathbf{1 \to \overline{1}}$ один раз, а при дворазовому натисканні **button** відбувається цикл $(\mathbf{1 \to \overline{1}})$ x **2**. Скетч має наступний вигляд.

```
int led1 = 10;
int led2 = 9;
int button = 8;
bool state = false;
void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(button, INPUT);
  Serial.begin(9600);
}
void loop() {
  if (digitalRead(button) == HIGH)
/*умова реалізації циклу 1-N1 при першому натисканні
кнопки button*/
  {
    state = true;
    digitalWrite(led1, HIGH);
  delay(1000);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(1000);
```

```
  digitalWrite(led2, LOW);
    }
    Serial.println(state);
    if (digitalRead(button) == HIGH && state == true)
/*умова реалізації циклу(1-N1)x2 при другому
натисканні кнопки button*/
  {
    state = false;
    digitalWrite(led1, HIGH);
  delay(1000);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(1000);
  digitalWrite(led2, LOW);
  delay(200);
  digitalWrite(led1, HIGH);
  delay(1000);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(1000);
  digitalWrite(led2, LOW);
  }
}
```

9.2. **Розробка проекту з контролером для виконання циклу $\mathbf{1-\overline{1}}$ з БІстабільним керуванням та одним кінцевим вимикачем**

Мета проекту: отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $\mathbf{1-\overline{1}}$ з БІстабільним керуванням та одним кінцевим вимикачем.

Основов'язальні машини мають багато важільний цикловий механізм прокачки гребінки вушкових голок з функцією вистою веденої ланки.

Для комп'ютерно-інтегрованих технологічних машин легкої промисловості можливо спрощення кінематики таких механізмів за рахунок забезпечення програмування циклу роботи 4-ланкового

механізму замість існуючих багато важільних механізмів машин легкої промисловості.

В машинах та автоматизованих системах легкої промисловості можуть виникнути умови коли контроль вихідної ланки по положенню є недоцільним чи не можливим. До таких умов належать:

- обмеження по габаритам, коли датчик занадто великий і його неможливо розмістити не порушуючи конструкції механізму чи коли механізм занадто малий в порівнянні з датчиком;
- наявність електромагнітних полів в зоні можливого розташування датчика, які можуть створювати шуми, що можуть призвести до не пропускання сигналу від датчику, або навпаки хибного чи несвоєчасного спрацювання датчика;
- економічна недоцільність, коли механізм виконує не складну операцію в легких умовах і замість датчика можна задати витримку часу за допомогою контролера, якої точно вистачить для переходу вихідної ланки механізму в необхідну позицію.

Технічне завдання. Розробити програму керування механізмом прокачки гребінки вушкових голок основов'язальних машин з автоматичним контролем втягнутого положення по кінцевому вимикачу, а витягнутого положення по часу. Час витягнутого положення штоку поршню складає 3 секунди. Вмикання системи при натисненні кнопки S1, зупинка при відпрацюванні циклу після натиснення кнопки S2.

Вибираємо наступне обладнання: контролер Festo FC34; пневматичний циліндр двосторонньої дії з БІстабільним пневматичним розподільником 5/2 з електромагнітним керуванням (електромагніти *Y1* та *YN1*); два пневматичні дроселі зі зворотними клапанами; кінцевий вимикач *XN1* для втягнутого положення штоку; кнопку «Start» S1 і кнопку «Stop» S2; електромагнітні реле *K1* і *K2*; джерело постійного струму DC 24 В.

На рис. 9.7 наведена схема з програмованим контролером для виконання циклу 1 → N1 та одним кінцевим вимикачем.

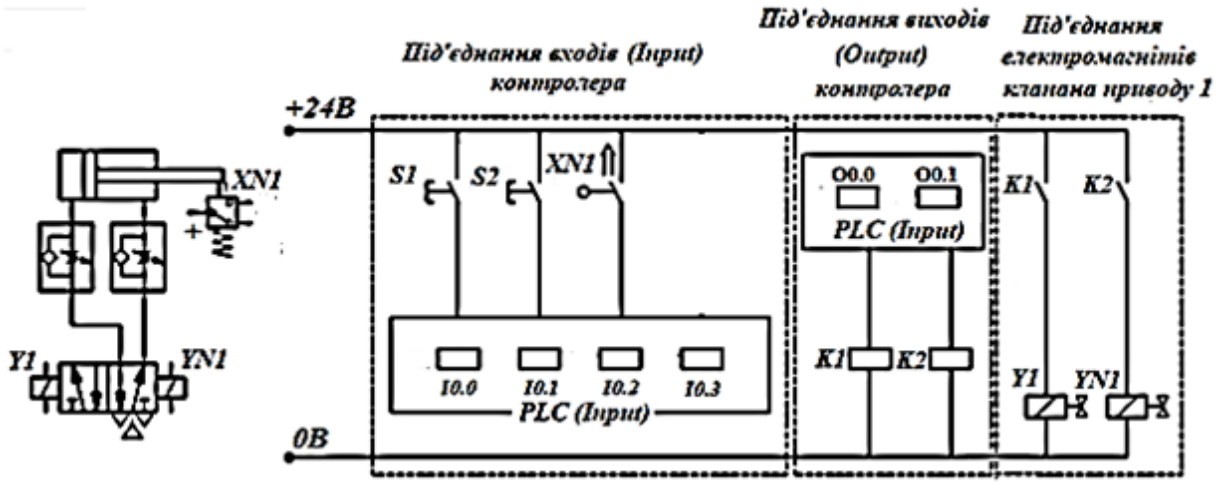


Рис. 9.7. Комбінована схема з контролером для виконання циклу $1-\overline{1}$ з БІстабільним керуванням та одним кінцевим вимикачем

Allocation List для програми складаємо з урахуванням таблиці 9.1.

Складання логічних команд керування. Система буде працювати за циклом $1-\overline{1}$. Але одночасно з подачею сигналу *Y1* ми будемо вмикати таймер *T1* і коли включений таймер відрахує встановлену затримку часу, необхідно буде подати сигнал *YN1* на відведення штоку поршню циліндру в початкове положення. Після включення таймеру необхідно буде обов'язково увімкнути елемент пам'яті *EP1*. Оскільки другий кінцевий вимикач відсутній тому сигналу *X1* немає і його фактично замінюють сигнал вимкнення таймера *NT1* та сигнал включення елемента пам'яті *EP1*. Граф циклу наведений на рис. 9.8.

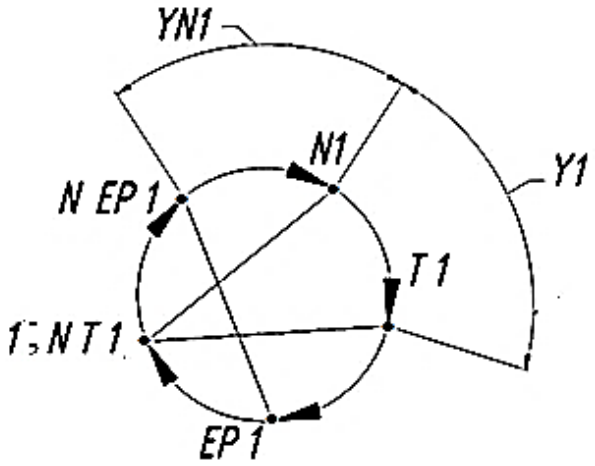


Рис. 9.8. Граф циклу $\mathbf{1-\overline{1}}$ з одним кінцевим вимикачем *XN1* для контролю втягнутого положення штоку поршня пневмоциліндру і контролем другого положення по сигналам *NT1* і *EP1*

Записуємо систему рівнянь причино-наслідкових зв'язків у відповідності до графу на рис.9.12 при БІстабільному керуванні за схемою на рис. 9.11:

$$Y1 \Leftarrow XN1 \cdot NEP1 \cdot XST \quad (9.5)$$

$$T1 \Leftarrow XN1 \cdot NEP1 \cdot XST \quad (9.6)$$

$$EP1 \Leftarrow T1 \quad (9.7)$$

$$NEP1 \Leftarrow NT1 \quad (9.8)$$

$$YN1 \Leftarrow NEP1 \quad (9.9)$$

Включення елемента пам'яті таймера допустиме в одній команді з самим таймером тому:

$$EP1 \Leftarrow NEP1 \cdot XST \quad (9.10)$$

Створення програми у середовищі FST4.21 за логічними командами графу на рис. 9.8 та рівняннями причино-наслідкових зв'язків (9.5)...(9.10).

```
'Підготовка системи до пуску. Приведення приводів у
'початковий стан та вимикання елемента пам'яті EP1,


STEP 0
 IF                    NOP
 THEN  RESET           Y1
       SET             YN1
       RESET           XST
       RESET           EP1

'Перехід до STEP 1 після переведення системи в
'початковий стан

IF                    XN1
THEN  JMP TO 1
```

```
'В робочому кроці 1 для початку роботи необхідно
'натиснути кнопку «Start» S1 за умови, що система
'перебуває в початковому положенні і не натиснена
'кнопка «Stop» S2, яка відповідає за зупинку циклу.
'При виконанні цих умов вмикаємо елемент пам'яті XST.

STEP 1
 IF                  S1
       AND     N     S2
 THEN  SET           XST

'Далі записуємо складені команди. Важливо пам'ятати,
'що для приводів з БІстабільним керуванням необхідно
'в одній команді вмикати один сигнал і вимикати
'інший. Також бажано для зручності записувати команди
'так, як вони йдуть в циклі

IF                   XST
       AND           XN1
       AND     N     EP1
 THEN  SET           T1
         WITH        3s
       SET           Y1
       RESET         YN1
       SET           EP1
 IF            N     T1
       AND           EP1
 THEN  RESET         EP1
 IF            N     EP1
 THEN  SET           YN1
       RESET         Y1

'Також в програмі необхідно передбачити зупинку
'циклу. В нашому випадку кнопка S2 вимкне елемент
'пам'яті XST і система зупиниться після закінчення
'циклу та буде чекати повторної команди початку
'роботи.
```

```
IF                    S2
THEN   RESET          XST

'Останній крок програми необхідно завершувати
'командою на переведення системи в інший або в цей же
'крок

IF                    NOP
THEN  JMP TO 1
```

Для порівняльного аналізу на рис. 9.9 наведена комбінована схема без контролера для циклу $1-\bar{1}$ з одним кінцевим вимикачем і БІстабільним керуванням, яка відрізняється від функціонально адекватної схеми на рис. 8.16 у розділі 8.

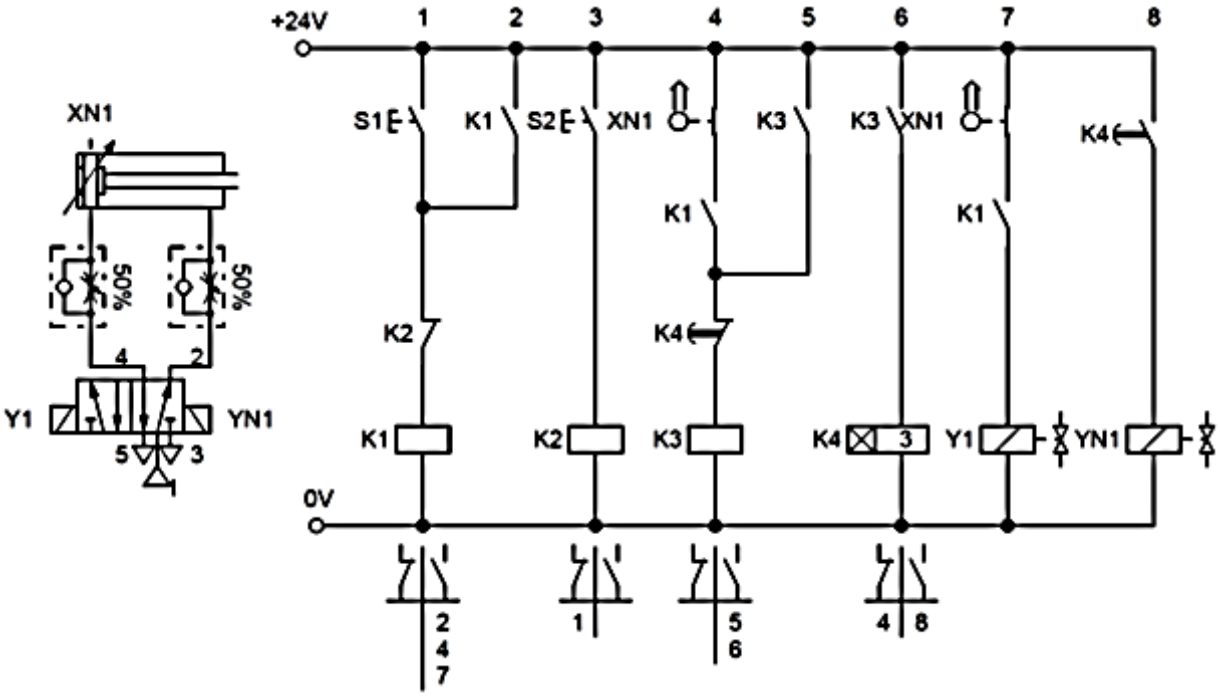


Рис. 9.9. Комбінована схема без контролера для виконання циклу $1-\bar{1}$ з БІстабільним керуванням та одним кінцевим вимикачем

Завдання для самостійної роботи:

1. Переробити програму для проекту з контролером для виконання циклу $1-\bar{1}$ з МОНОстабільним керуванням та одним кінцевим

вимикачем. Прийняти робочий хід пневмоциліндру 1 секунда, а також забезпечити робочий хід циліндра тільки при затисненій кнопки S1.

2. Порівняти комбіновану схему з контролером (рис. 9.7) з комбінованою схемою без контролера (рис.9.9) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

9.3. **Розробка проекту з контролером для виконання циклу $1-\bar{1}$ з БІстабільним керуванням та без кінцевих вимикачів**

Мета проекту отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $1-\bar{1}$ без кінцевих вимикачів для механізмів з періодичної функцією і вистоєм веденої ланки за аналогом багатоважільних механізмів для комп'ютерно-інтегрованих технологічних машин легкої промисловості.

Технічне завдання. Розробити програму керування механізмом подачі заготовки в зоні її обробки з контролем втягнутого та витягнутого положення по часу. Достатній час для висування циліндру в зону обробки – 4с. Зворотній хід поршню виконувати за 2 с. Вмикання системи при натисненні кнопки S1, зупинка після натиснення кнопки S2 і відпрацювання циклу.

Вибираємо наступне обладнання:

контролер Festo FC34; пневматичний циліндр двосторонньої дії з БІстабільним пневматичним розподільником 5/2 з електромагнітним керуванням (електромагніти Y1 і YN1); два пневматичні дроселі зі зворотними клапанами; кнопку «Start» *S1* і кнопку «Stop» *S2*; електромагнітні реле К1 і К2; блок живлення постійним струмом (DC) 24 В.

На рис. 9.10 наведена комбінована схема з контролером і без кінцевих вимикачів для виконання з циклу $1-\bar{1}$ з БІстабільним керуванням.

Allocation List потрібно скласти з урахуванням таблиці 9.1.

Складання логічних команд керування: система буде працювати за циклом $1-\bar{1}$. Але одночасно з подачею сигналу ***Y1*** ми будемо програмно вмикати таймер ***T0*** і коли ввімкнений таймер відрахує 4 секунди, необхідно буде подати сигнал ***YN1*** на відведення поршню циліндру в початковий стан і також буде програмно включеним таймер ***T1*** з

витримкою в часі на 2 секунди. Після включення таймеру ***T0*** необхідно буде обов'язково включити елемент пам'яті ***EP1***. Після включення таймеру ***T1*** необхідно буде обов'язково увімкнути ще один елемент пам'яті, але можна просто вимкнути елемент пам'яті ***EP1***. Граф циклу наведений на рис. 9.11.

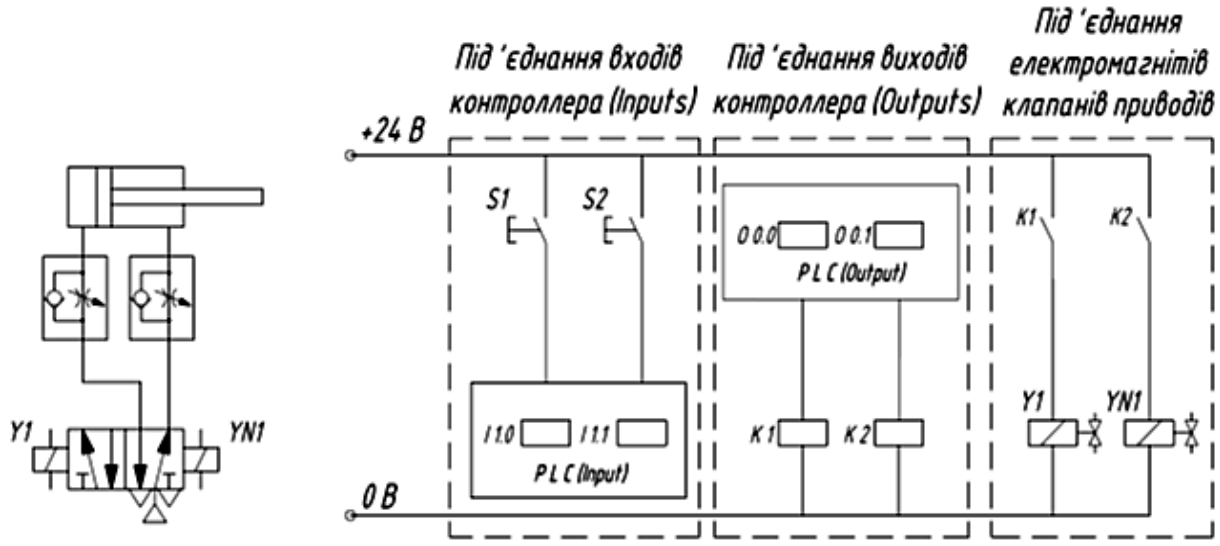


Рис. 9.10. Комбінована схема з контролером і без кінцевих вимикачів для виконання з циклу $1-\bar{1}$ з БІстабільним керуванням

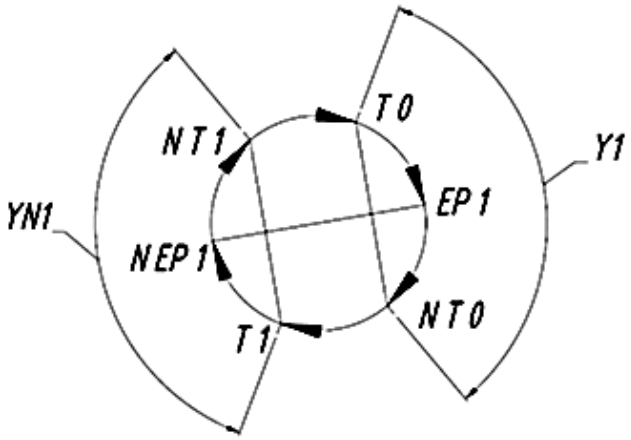


Рис. 9.11. Граф циклу $1-\bar{1}$ з двома таймерами T0 і T1, одним елементом пам'яті EP1 і без кінцевих вимикачів

Записуємо наступну систему рівнянь причино-наслідкових зв'язків (логічних команд) для виконання з циклу $1-\bar{1}$ з БІстабільним керуванням у відповідності до схеми на рис. 9.9 та графу на рис. 9.11:

$$Y1 \Leftarrow NT1 \cdot NEP1 \cdot XST \quad (9.11)$$

$$T0 \Leftarrow T1 \cdot EP1 \cdot XST \quad (9.12)$$

$$EP1 \Leftarrow T0 \quad (9.13)$$

$$YN1 \Leftarrow NT0 \cdot EP1 \quad (9.14)$$

$$T1 \Leftarrow NT0 \cdot EP1 \quad (9.15)$$

$$NEP1 \Leftarrow T1 \quad (9.16)$$

Включення та вимикання елемента пам’яті таймера допустиме в одній команді з самим таймером тому:

$$EP1 \Leftarrow NT1 \cdot NEP1 \cdot XST \quad (9.17)$$

$$NEP1 \Leftarrow \mathrm{N}T0 \cdot EP1 \quad (9.18)$$

Створення програми у середовищі FST4.21 по складеним логічним командам (9.11)… (9.18):

```
‘Підготовка системи до пуску. Приведення приводів у
‘початкове положення та вимикання елементів пам’яті,
‘які використовуються в програмі та перехід в крок 1

STEP 0
 IF                    NOP
 THEN  RESET           Y1
       SET             YN1
       RESET           XST
       RESET           EP1
   JMP TO 1

‘В робочому кроці 1 для початку роботи необхідно
‘натиснути кнопку «Start» S1 за умови, що система
```

```
'перебуває в початковому положенні і не нетиснена
'кнопка «Stop» S2, яка відповідає за зупинку циклу.
'При виконанні цих умов вмикаємо елемент пам'яті
'XST, який був використаний при складанні команд
'керування

STEP 1
 IF                  S1
       AND     N     S2
 THEN  SET           XST

'Далі записуємо складені команди. Важливо пам'ятати,
'що при керуванні приводами з БІстабільним керуванням
'необхідно в одній команді вмикати один сигнал і
'вимикати інший. Також бажано для зручності
'записувати команди так, як вони йдуть в циклі

 IF            N     T1
       AND     N     EP1
       AND           XST
 THEN  SET           T0
         WITH        4s
       SET           EP1
       SET           Y1
       RESET         YN1
 IF            N     T0
       AND           EP1
 THEN  SET           T1
         WITH        2s
       RESET         EP1
       SET           YN1
       RESET         Y1

'Також в програмі необхідно передбачити зупинку
системи. 'В нашому випадку кнопка S2 вимкне елемент
пам'яті XST і 'система зупиниться після закінчення
циклу та буде 'чекати повторної команди початку
роботи
```

```
IF                    S2
THEN  RESET           XST

'Останній крок програми необхідно завершувати коман-
'дою на переведення системи в інший або в цей же крок

IF                    NOP
THEN  JMP TO 1
```

Для порівнювального аналізу на рис. 9.12 наведена комбінована схема без контролера і без кінцевих вимикачів для виконання з циклу $1-\bar{1}$ з БІстабільним керуванням. Ця схема відрізняється від функціонально адекватної схеми на рис. 8.24 у розділі 8.

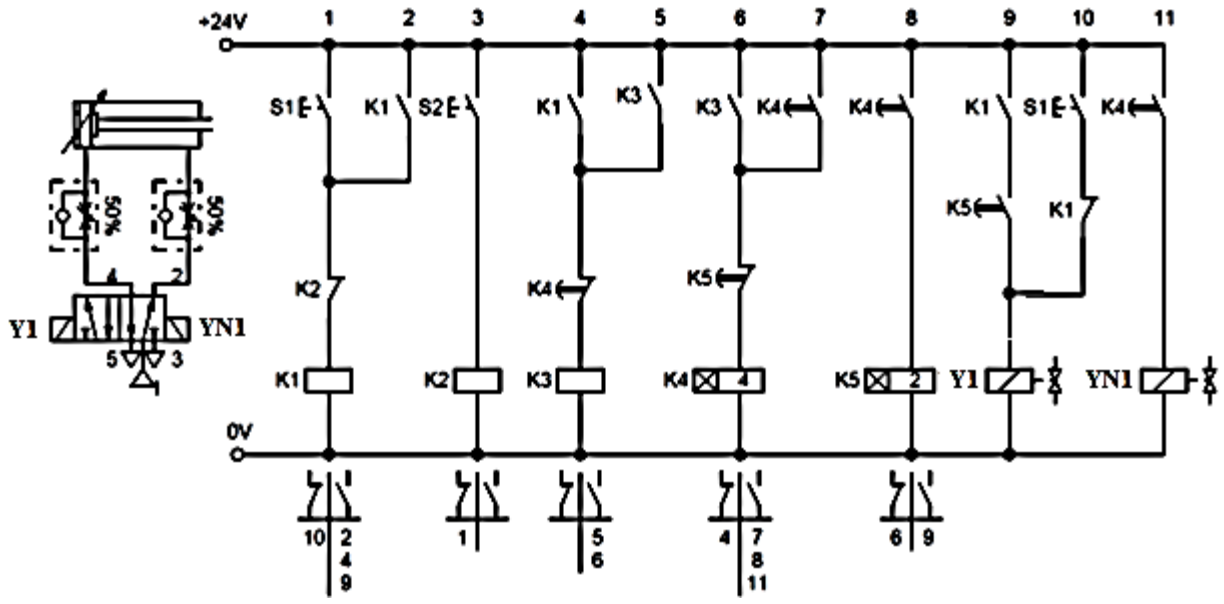


Рис. 9.12. Комбінована схема без контролера і без кінцевих вимикачів для виконання з циклу $1-\bar{1}$ з БІстабільним керуванням

Завдання для самостійної роботи:

1. Зробити програму для керування системою з аналогічним циклом, але з МОНОстабільними розподільником. Забезпечити преселектором таймера робочий і зворотній хід поршню циліндру 3 секунди.

2. Порівняти комбіновану схему з контролером (рис. 9.10) з комбінованою схемою без контролера (рис. 9.12) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

9.4. **Розробка проекту з контролером для виконання циклу $1-\overline{1}$ з БІстабільним керування, без кінцевих вимикачів та одночасним натисненням двох кнопок «Start»**

Мета проекту: отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $1-\overline{1}$ керування промисловим пресом вирубання деталей для виготовлення взуття. Вмикання пресу забезпечити тільки при одночасним натисненням двох кнопок «Start» для захисту від можливих травм оператора.

Одним з найрозповсюдженіших порушень правил безпеки операторами при роботі з пресовим обладнанням є притримування заготовки в зоні пресування при безпосередній роботі преса. Існує стандартний захист оператора при роботі з пресовим обладнанням – пресова подушка почне опускатись після натиснення двох кнопок, які розташовані на відстані 1 м одна від одної, для того, щоб під час пускання пресової подушки обидві руки оператора знаходились поза робочої зони пресування. Необхідність натиснення двох кнопок оператором можна обійти наступним чином: одна кнопка буде затиснена заздалегідь важким предметом, а після цього оператор може притримувати заготовку в зоні пресування та іншою рукою тиснути на другу кнопку.

Для уникнення такого порушення прес повинен відпрацьовувати тільки при одночасному двох натисненні кнопок. У випадку відтиснення хоча б однієї з кнопок подушка преса повинна повернутись в початкове положення. Наступний робочий хід подушки преса відбудеться тільки після відтиснення двох кнопок і наступного одночасного їх натиснення.

Технічне завдання. Розробити програму керування пресом, яка забезпечує опускання пресу тільки у випадку одночасного натиснення двох кнопок. При відпусканні хоча б однієї кнопки прес повертається в початкове положення. Повторне включення пресу можливе тільки

повернені пресу в початкове положення (перед одночасним натисненням двох кнопок вони повинні бути відтиснені).

Вибираємо наступне обладнання: PLC Festo FC34; пневматичний циліндр двосторонньої дії з БІстабільним пневматичним розподільником 5/2 з електромагнітним керуванням (електромагніти *Y1* та *YN1*); два пневматичні дроселі зі зворотними клапанами; кнопки «*Start*» *S1* та «*Start*» *S2*; блок електромагнітних реле. На лабораторному стенді зібрати схему з контролером.

На рис. 9.13 наведена комбінована схема з контролером і без кінцевих вимикачів для виконання з циклу $1-\bar{1}$ з одночасним натисненням двох кнопок «Start» *S1* і *S2*.

Allocation List потрібно скласти з урахуванням таблиці 9.1.

Складання логічних команд керування: насправді одночасно натиснути дві кнопки неможливо. Тому на практиці надається невеликий інтервал часу в 0.5…1.0 секунду для натиснення двох кнопок. Логіка роботи преса наступна: при натисненні хоча б однієї з кнопок та вимкненому елементі пам'яті таймера вмикається таймер на 0.5 секунди та елемент пам'яті для таймера.

$$\boldsymbol{TO} \Leftarrow (\boldsymbol{S1}+\boldsymbol{S2})\cdot \boldsymbol{NEP1} \tag{9.19}$$

$$\boldsymbol{EP1} \Leftarrow (\boldsymbol{S1}+\boldsymbol{S2})\cdot \boldsymbol{NEP1} \tag{9.20}$$

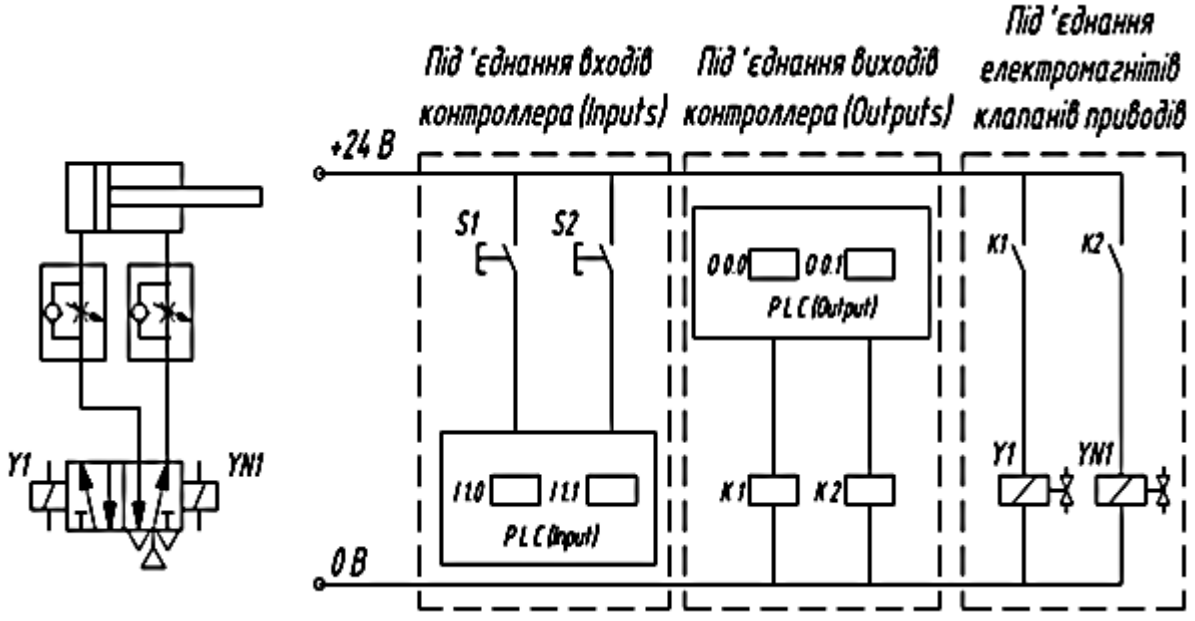


Рис. 9.13. Комбінована схема з контролером і без кінцевих вимикачів для виконання з циклу $1-\bar{1}$ з одночасним натисненням двох кнопок «Start»

Далі задаємо умову руху пресової подушки - натисненні дві кнопки та працюючий таймер:

$$\boldsymbol{Y1 \Leftarrow S1 \cdot S2 \cdot T0} \tag{9.21}$$

При відтисненій хоча б одній з кнопок - повернути подушку преса в верхнє положення:

$$\boldsymbol{YN1 \Leftarrow N\,S1 + NS2 \cdot} \tag{9.22}$$

Також для приведення системи в початкове положення необхідно вимкнути елемент пам'яті таймера. Вимикаємо елемент пам'яті таймера при відтиснені двох кнопок:

$$\boldsymbol{NEP1 \Leftarrow N\,S1 \cdot NS2 \cdot} \tag{9.23}$$

Створення програми в FST4.21 по складеним логічним командам:

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення, вимикання елементF пам'яті EP1
'та перехід до кроку 1

STEP 0
 IF                    NOP
 THEN  SET             YN1
       RESET           Y1
       RESET           EP1
       JMP TO 1

'В робочому кроці записуємо раніше складені логічні
'команди керування

STEP 1
 IF                  ( S1
       OR              S2 )
```

```
       AND      N       EP1
 THEN  SET              T0
         WITH           0.5s
       SET              EP1
 IF                     S1
       AND              S2
       AND              T0
 THEN  SET              Y1
       RESET            YN1
 IF              N      S1
       OR        N      S2
 THEN  SET              YN1
       RESET            Y1
 IF              N      S1
       AND       N      S2
 THEN  RESET            EP1

'Останній крок програми необхідно завершувати
'командою на переведення системи в крок 1

IF                      NOP
THEN  JMP TO 1
```

Для порівнювального аналізу на рис. 9.14 наведена комбінована схема без контроленра для виконання з циклу $1-\bar{1}$ з одночасним натисненням двох кнопок

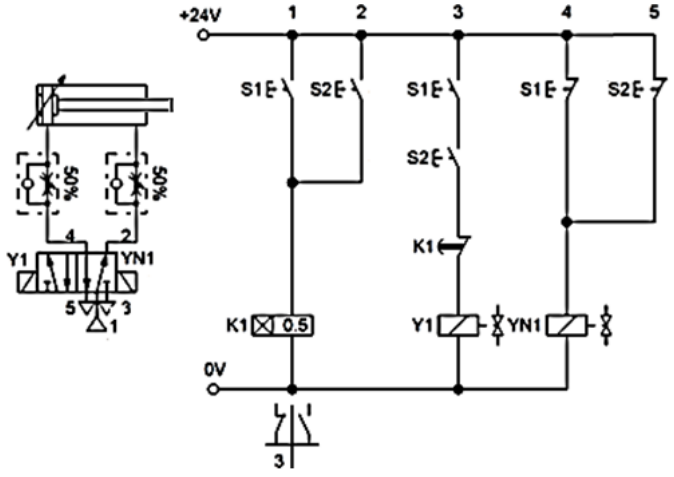


Рис. 9.14. Комбінована схема без контролера для виконання з циклу $1-\bar{1}$ з одночасним натисненням двох кнопок S1 і S2

Завдання для самостійної роботи:

1. Розробити подібну систему для керування пресом з МОНОстабільним керуванням із спрацюванням пресу при натисненні на кнопок S1, S2 та S3 з інтервалом не більше 1 с.

2. Порівняти комбіновану схему з контролером (рис. 9.17) з комбінованою схемою без контролера (рис. 9.18) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

9.5. Розробка проекту з використання в логічних командах Булевої алгебри для циклу $(S1 \cdot S2) \to 1 \to (S1 + S2) \to \overline{1}$

Мета проекту: отримання практичних навичок по програмуванню циклу $(S1 \cdot S2) \to 1 \to (S1 + S2) \to \overline{1}$ для роботи вирубних пресів деталей верху і низу взуття і пресів для вирубання з настилу дрібних деталей крою з текстилю для швейних виробів. При цьому існують підвищенні вимоги до техніки безпеки при роботі на такому технологічному обладнані. А саме, коли оператор пресу подає різак в зону пресування або знімає його з вирубальної плити, необхідно щоб руки оператора не знаходились в зоні пресування під час опускання подушки пресу. Тому команда (сигнал) на опускання подушки пресу повинна подаватися одночасним натисненням двох кнопок «Start», які розташовані на достатній відстані одна від другої.

Логічні функції, які часто використовують у таких програмах мехатронних циклових систем з логічними операціями засновані на алгебрі логіки, початок якій покладено працями англійського математика Дж. Буля і тому її також називають Булевою алгеброю або алгеброю виразів. Булева алгебра – це частина математики, яка вивчає алгебру логіки, а саме застосування алгебраїчних методів і символіки для вивчення логічних відношень і розв'язання формалізованих логічних задач. Істинному висловленню приписується стан «1», хибному – стан «0». Висловлення можуть бути простими і складними. Складні висловлення складаються з простих.

Для об'єднання простих висловлень в складні використовуються логічні зв'язки, що відповідають логічним функціям, аргументами яких є прості висловлення.

Взагалі, вся Булева алгебра базується на наступних аксіомах для логічної змінної X:

$\bar{\bar{X}} = X$; $X + \bar{X} = 1$; $X + 1 = 1$; $X + X = X$; $X + 0 = X$;

$X \cdot \bar{X} = 0$; $X \cdot X = X$; $X \cdot 0 = 0$; $X \cdot 1 = X$.

Уявимо, що циклова система керування має два дискретні входи X1 та X2 і вихідний дискретний сигнал Y1, який є логічною функцією від входів ($Y1 = f(X1, X2)$). Розглянемо три випадки залежності $Y1 = f(X1, X2)$.

В першому випадку необхідно, щоб сигнал Y1=1 при наявності хоча б одного з сигналів X1=1 чи X2=1. Така логічна залежність реалізується логічним додаванням, поширена назва логічна операція «***диз'юнкція***», яка у програмах позначається оператором **OR** (АБО): $Y1 = X1 + X2$.

Для оператора **OR** таблиця 9.3 істинності сигналів має наступний вигляд:

Таблиця 9.3 Істинність сигналів для логічного складання

X1	X2	*Y1:= X1* ***OR*** *X2 = X1* $\vee$ *X2*
0	**0**	**0**
1	**0**	**1**
0	**1**	**1**
1	**1**	**1**

В другому випадку необхідно, щоб сигнал Y1=1 був при наявності одночасно сигналів X1 та X2. Така логічна залежність реалізується логічним множенням, поширена назва «***кон'юнкція***», яка в програмах позначається оператором **AND** (*I*):

$Y1 = X1 \cdot X2$.

Для оператора **AND** таблиця 9.4 істинності сигналів керування має наступний вигляд:

Таблиця 9.4 Істинність сигналів для логічного множення

X1	X2	*Y1:= X1* ***AND*** *X2 = X1* $\wedge$ *X2*
0	**0**	**0**
1	**0**	**0**
0	**1**	**0**
1	**1**	**1**

В третьому випадку необхідно, щоб сигнал на виході Y1=1 при відсутності сигналу на вході X1=0 і навпаки Y1 = 0 при X1 = 1. Така логічна залежність реалізується інверсією і в програмах позначається оператором **N** (ні): $Y1 = \overline{X1}$.

Для оператора **N** таблиця 9.5 істинності сигналів керування має наступний вигляд:

Таблиця 9.5 Істинність сигналів для логічної операції «інверсія»

X1	$Y1 = \overline{X1}$
0	**1**
1	**0**

Реалізація логічних операцій за допомогою мови STL

В програмному рядку логічне множення реалізується для оператора AND наступним чином:

```
X1 AND X2.
```

В програмному рядку логічне додавання реалізується для оператора **OR** наступним чином:

```
X1 OR X2.
```

У випадку якщо логічна змінна **X1** чи **X2** складаються з декількох змінних, то їх треба взяти в дужки:

```
(X1 AND X2) OR (X1 OR X2)
```

В програмному рядку логічна операція інверсія реалізується для оператора **N** наступним чином:

```
N X1.
```

Технічне завдання. Виконати проект циклової системи керування вирубним пресом. Програма повинна забезпечувати опускання рухомої плити пресу на різак тільки у випадку одночасного натиснення двох кнопок «Start» S1 і S2. При відпусканні хоча б однієї кнопки прес повертається в початкове положення. Повторне включення пресу можливе тільки після повернення пресу в початкове положення (перед одночасним натисненням двох кнопок вони повинні бути відтиснуті).

Вибираємо наступне обладнання: контролер *Festo FC34* (Німеччина); пневматичний циліндр двосторонньої дії; БІстабільний пневматичний розподільник 5/2 з електромагнітним керуванням (електромагніти Y1 та YN1); два пневматичні дроселі зі зворотними клапанами; два кінцевих вимикача з електричними контактами XN1 та X1; кнопки «Start» S1 та «Пуск» S2; електромагнітні реле *K1* і *K2*; джерело живлення постійним струмом (***DC*** – ***D****irect* ***C****urrent*) 24 В.

На лабораторному стенді збираємо схему з контролером (рис. 9.19) для виконання циклу. $(\boldsymbol{S1 \cdot S2}) \rightarrow \mathbf{1} \rightarrow (\boldsymbol{S1 + S2}) \rightarrow \overline{\mathbf{1}}$.

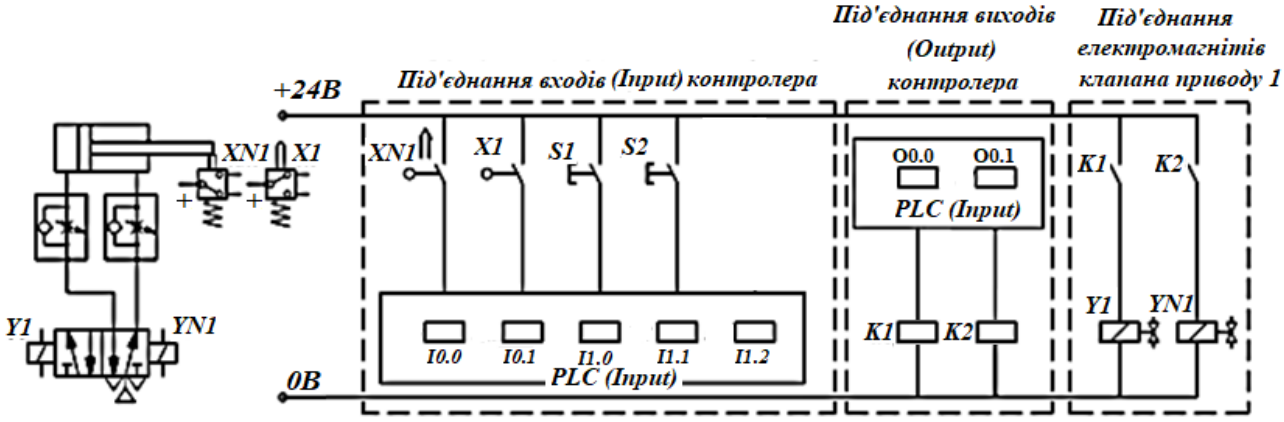


Рис. 9.19. Комбінована схема *з контролером* для виконання циклу $(\boldsymbol{S1 \cdot S2}) \rightarrow \mathbf{1} \rightarrow (\boldsymbol{S1 + S2}) \rightarrow \overline{\mathbf{1}}$

Allocation List потрібно скласти з урахуванням таблиці 9.1 і рис. 9.19.

Складання логічних команд керування: включення пресу буде відбуватись тільки по натисненні двох кнопок:

$$\mathbf{Y1} \Leftarrow \mathbf{S1 \cdot S2} \qquad (9.5)$$

В мехатронній системі керування з контролером передбачений елемент пам'яті *EP1*, який запам'ятовує, що ударна плита вирубного пресу опустилася у ніжне положення і при цьому датчик X1 кінцевого положення буде замкнений:

$$\boldsymbol{EP1} \Leftarrow \boldsymbol{X1} \qquad (9.6)$$

Повернення рухомий плити пресу в початкове положення відбудеться при відтисненні хоча б однієї кнопки S1 або S2:

$$\boldsymbol{Y1 \Leftarrow \overline{S1} + \overline{S2}} \tag{9.7}$$

Для уникнення порушення правил безпеки оператором, а саме затиснення однієї з кнопок вантажем чи іншім чином, для повторного опускання рухомий плити пресу, необхідно щоб вона повернулася в початкове положення і були відтисненні обидві кнопки S1 і S2. Тоді вимкнеться елемент пам'яті *EP1*, обернений сигнал якого входить в команду робочого ходу преса. Елемент пам'яті вимикається тільки коли рухома плита пресу повертається в початкове положення і дві кнопки віджаті, тобто:

$$\boldsymbol{\overline{EP1} \Leftarrow XN1 \cdot \overline{S1} \cdot \overline{S2}} \tag{9.8}$$

Новий робочий хід рухомий плити пресу повинен виконуватись тільки при вимкненому елементі пам'яті, тому команду (9.5) робочого руху плити пресу переписуємо у наступному вигляді:

$$\boldsymbol{Y1 \Leftarrow S1 \cdot S2 \cdot \overline{EP1}} \tag{9.9}$$

Створення програми в FST4.21 по складеним логічним командам (9.5)…(9.9):

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
пам'яті, 'які використовуються в програмі
STEP 0
IF                      NOP
THEN     SET            YN1
         RESET          Y1
         RESET          EP1

'Перехід до STEP 1 після переведення системи в
початковий 'стан
IF                   XN1
THEN     JMP TO 1
```

```
'В робочому кроці записуємо раніше складені команди

STEP 1
IF                    S1          ' вираз(9.9)
       AND            S2
       AND     N      EP1
THEN   SET            Y1
       RESET          YN1
IF                    X1          '(вираз(9.6)
THEN   SET EP1
IF             N      S1          '(вираз(9.7)
       OR      N      S2
THEN   SET            YN1
       RESET          Y1
IF             N      S1          '(вираз(9.8)
       AND     N      S2
       AND           XN1
THEN   RESET          EP1

'Останній крок програми необхідно завершувати
командою на 'переведення системи в цей же крок STEP 1

IF     NOP
THEN   JMP TO 1
```

Для порівняльного аналізу на рис. 9.20 наведена комбінована схема без контролера для циклу $(\boldsymbol{S1}\cdot\boldsymbol{S2}) \to \mathbf{1} \to (\boldsymbol{S1}+\boldsymbol{S2}) \to \overline{\mathbf{1}}$ з БІстабільним керуванням.

Завдання для самостійної роботи:

1.Розробити систему керування вирубним пресом для циклу $(\boldsymbol{S1}\cdot\boldsymbol{S2}) \to \mathbf{1} \to (\boldsymbol{S1}+\boldsymbol{S2}) \to \overline{\mathbf{1}}$ з МОНОстабільним керуванням, початок роботи якої неможливий без затиснення кнопки з фіксатором S3.

2. Порівняти комбіновану схему з контролером (рис. 9.19) з комбінованою схемою без контролера (рис. 9.20) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

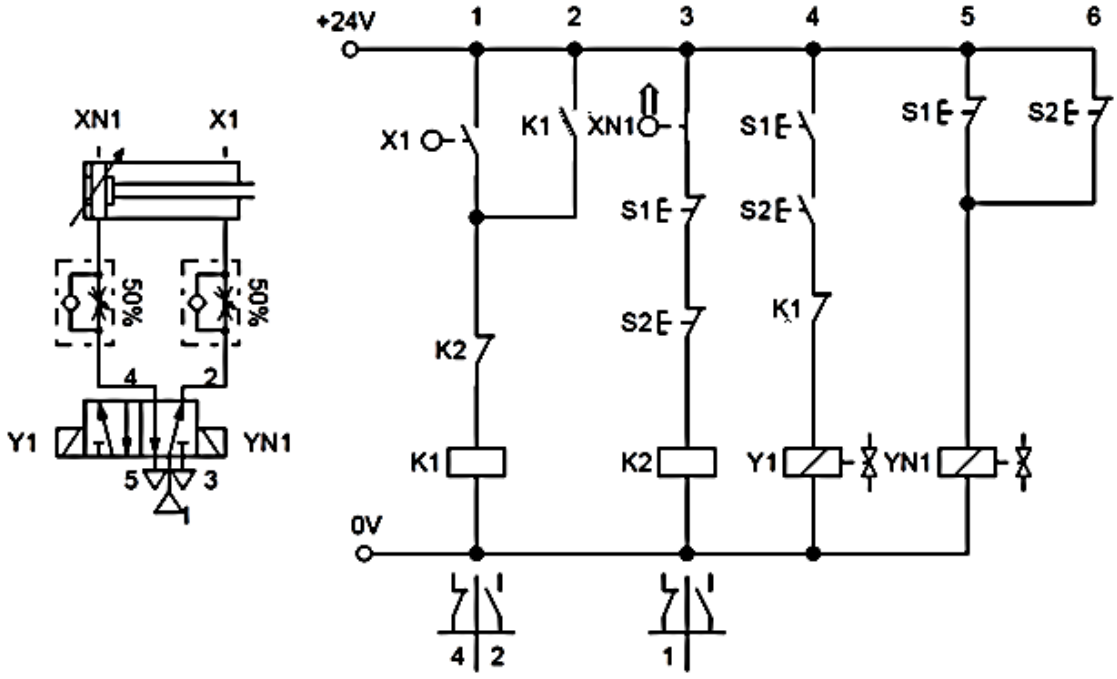


Рис. 9.20. Схема без контролера для циклу $(\boldsymbol{S1 \cdot S2}) \rightarrow \mathbf{1} \rightarrow (\boldsymbol{S1 + S2}) \rightarrow \overline{\mathbf{1}}$

9.6. **Розробка проекту з контролером для циклу** 1-TIMER-N1 **з БІстабільним керуванням та двома кінцевими вимикачами**

Мета проекту: отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $1 - 5\text{c} - \overline{1}$ із затримкою на заданий час за допомогою таймера**.**

В автоматизованих системах машин легкої промисловості часто виникають задачі, коли необхідно, щоб робочий орган машини після виходу на робочу позицію перебував на ній протягом певного заданого проміжку часу (наприклад у випадку приклеювання підошви до заготовки черевика та ін.). В таких випадках необхідно використовувати *реле часу* (при складанні системи на базі реле та їх контактів) та *таймеру* (при створенні програми при роботі з контролером). При цьому потрібне програмування циклу виконання періодичної функцією і вистоюванням веденої ланки за аналогом багато важільних механізмів для комп'ютерно-інтегрованих технологічних машин легкої промисловості.

При створенні програми роботи системи з витримкою часу важливо зробити такі умови, щоб таймер вмикався тільки необхідну кількість разів під час циклу і уникав повторного хибного вмикання.

Технічне завдання. Розробити проект керування пресом для приклеювання підошви черевика, забезпечивши витримку 5 секунд в нижньому положенні подушки преса. Вмикання системи при натисненні кнопки *S1*, зупинка після відпрацювання циклу – натиснення кнопки *S2*.

Вибираємо наступне обладнання: контролер Festo FC34; пневматичний циліндр двосторонньої дії з БІстабільним пневматичним розподільником 5/2 та електромагнітним керуванням (електромагніти Y1 та YN1); два пневматичні дроселі зі зворотними клапанами; два кінцевих вимикача з електричними контактами XN1 та X1; кнопка «Start»S1, кнопка «Stop» S2; електромагнітні реле *K1* і *K2*; джерело живлення постійним струмом (***DC*** – ***D****irect* ***C****urrent*) 24 В.

На рис. 9.21 наведена комбінована схема циклової системи для виконання циклу $\mathbf{1-5c-\overline{1}}$.

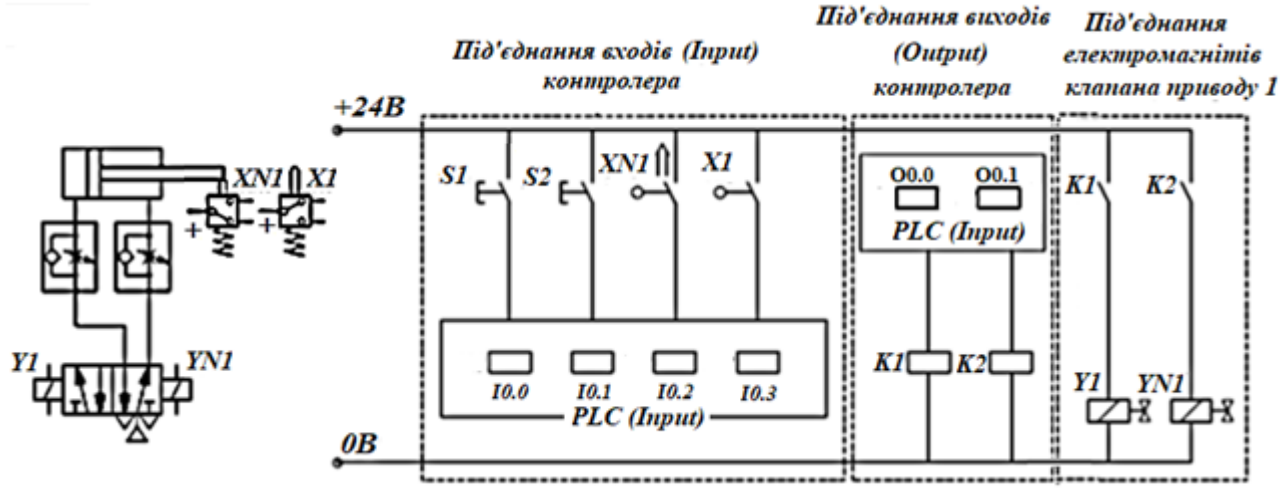


Рис. 9.21. Комбінована схема циклової системи для циклу $\mathbf{1-5c-\overline{1}}$

Allocation List потрібно скласти з урахуванням таблиці 9.1.

Складання логічних команд керування: мехатронна система працює за циклом $1 \to 5s \to \bar{1}$. Тобто при натисненні на датчик висунутого положення штока поршня пневмоциліндра повинен увімкнутися таймер на 5 с, після витримки часу таймер автоматично вимикається і подушка пресу повертається в верхнє початкове положення. Складемо граф циклу (рис. 9.22).

При аналізі графу (рис. 9.22) виявляється, що система керування інформаційно неповна, оскільки в графі можна провести одну лінію невизначеності, яка розділяє граф на два незалежних підграфа.

Прибираємо лінію невизначеності за допомогою введенням додаткового елементу пам’яті *EP* (рис. 9.23), де *EP1* команда включення елементу пам’яті, а *NEP1* команда на вимикання елемента пам’яті.

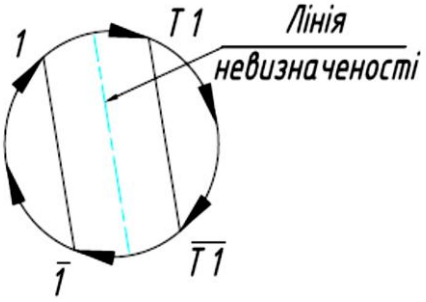


Рис. 9.22. Інформаційно невизначений граф циклу $1-5\text{с}-\overline{1}$ (без елемента пам’яті *EP1*)

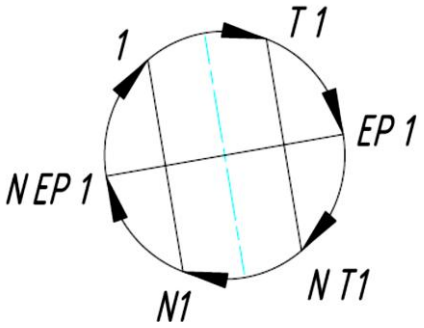


Рис. 9.23. Інформаційно визначений граф циклу $1-5\text{с}-\overline{1}$ (з елементом пам’яті *EP1*)

Як зазначалося раніше, таймер вимикається автоматично після відрахування заданого часу, тому зворотну команду для нього записувати не треба. Для спрощення в якості команди *EP1* на включення елементу пам’яті можна використати команду для включення таймера *T1*, оскільки ці два процеси відбуваються майже одночасно і на роботу системи не вплинуть, тобто $EP1 \Leftarrow X1 \cdot NEP1$.

Складання команд для вмикання таймера та опитування його відпрацювання при програмуванні контролера розглянемо на мові STL. В мові STL передбачені два варіанти задачі часу роботи таймера.

В першому варіанті просто прописується одноразова тривалість часу роботи таймера:

IF <умова>
THEN SET T1
WITH 5s…

Такий варіант завдання часу витримки зручне швидкістю написання та відсутністю додаткових операндів, можливістю використання одного і того ж таймеру в циклі для різних витримок часу, але недоліком є те що при багаторазовому вмикання одного і того ж таймера в циклі на однакову витримку треба кожного разу прописувати цю тривалість.

В другому варіанті додатково вводиться оператор преселектора часу таймера TPx (де x - номер, використаного таймера), який запам'ятовує заданий час витримки. Завантажити певне значення часу в *преселектор часу* можна в одній команді з вмикання таймеру, чи в попередніх командах, наприклад:

```
IF <умова>
THEN LOAD V500                    'тут  V500 це 5 секунд = 500 ·0,01 с
TO TP1 SET T1…
```

Зручність такого варіанту завдання часу витримки – при багаторазовому ввімкнені одного таймеру на однакову витримку її тривалість задається тільки один раз, недолік – введення додаткового операнда; для різної тривалості витримки необхідно використовувати декілька таймерів та їх преселекторів.

Приклад вмикання таймера та опитування його роботи та відпрацювання на мові STL:

```
'включення таймера T1
IF <умова> AND N EP1
THEN SET T1 WITH 3s SET EP1
'робота таймера T1
IF T1 AND EP1 THEN …
'відпрацювання таймера T1
IF N T1 AND EP1
THEN…
```

Створення програми в FST4.21 по складеним логічним командам, які наведені під графами на рис. 9.24:

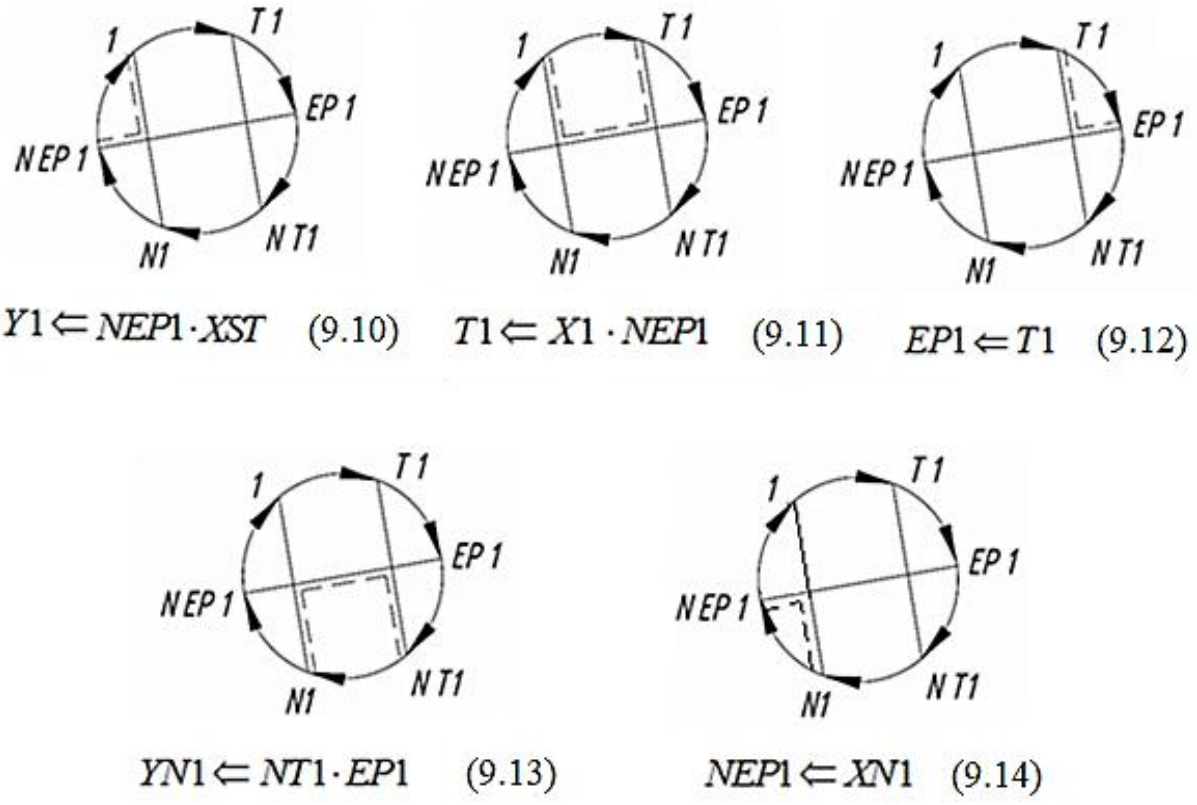


Рис. 9.24. Фрагменти графу при використані ліній зв'язку (пунктир) для побудові логічних команд керування циклу $1 - 5c - \overline{1}$

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
пам'яті, 'які використовуються в програмі

STEP 0
 IF                    NOP
 THEN  SET             YN1
       RESET           Y1
       RESET           XST
       RESET           EP1

'Перехід до STEP 1 після переведення системи в
початковий 'стан

IF            XN1
THEN  JMP TO 1
```

‘В робочому кроці № 1 для початку роботи необхідно натиснути ‘кнопку «Start» S1 за умови, що система перебуває в ‘початковому положенні і не нетиснена кнопка «Stop» S2, ‘яка відповідає за зупинку циклу. При виконанні цих умов ‘вмикаємо елемент пам’яті XST, який був використаний при ‘складанні команд керування

```
STEP 1
 IF                    S1
       AND     N       S2
       AND             XN1
 THEN  SET             XST
       RESET           EP1
```

‘Далі записуємо складені команди. Важливо пам’ятати, що ‘при керуванні приводами з БІстабільним керуванням ‘необхідно в одній команді вмикати один сигнал і вимикати ‘інший. Також бажано для зручності записувати команди ‘так, як вони йдуть в циклі

```
IF                    S1
       AND     N      S2
       AND            XN1
 THEN  SET            XST
       RESET          EP1
 IF            N      EP1
       AND            XST
 THEN  SET            Y1
       RESET          YN1
 IF                   X1
       AND     N      EP1
 THEN  SET            T1
         WITH         5s
       SET            EP1
 IF            N      T1
       AND            EP1
 THEN  SET            YN1
       RESET          Y1
```

```
 IF                    XN1
 THEN  RESET           EP1
'Також в програмі необхідно передбачити зупинку
системи. 'В нашому випадку кнопка S2 вимкне елемент
пам'яті XST і 'система зупиниться після закінчення
циклу та буде 'чекати повторної команди початку
роботи

IF                     S2
THEN  RESET            XST

'Останній крок програми необхідно завершувати
командою на 'переведення системи в інший або в цей же
крок

IF                     NOP
THEN  JMP TO 1
```

Для порівняльного аналізу на рис. 9.25 наведена комбінована схема без контролера для циклу $1-5c-\overline{1}$ з БІстабільним керуванням і двома кінцевими вимикачами.

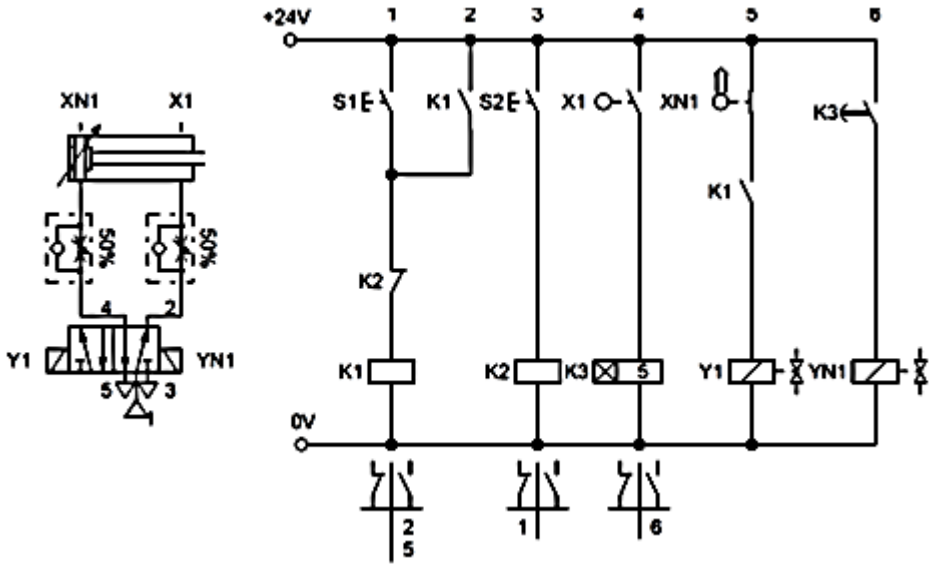


Рис. 9.25. Комбінована схема (тип С3) без контролера для виконання циклу $1-5c-\overline{1}$ з БІстабільним керуванням і двома кінцевими вимикачами

Завдання для самостійної роботи студентів:

1. Переробити програму для керування системою з аналогічним циклом, але з МОНОстабільним пневморозподільником. Забезпечити витримку часу 10 секунд у витягнутому положенні, а також забезпечити робочий хід циліндра ТІЛЬКИ при затисненій кнопки S1. Задача часу витримки через преселектор таймера.

2. Порівняти комбіновану схему *з контролером* (рис. 9.18) з комбінованою схемою *без контролера* (рис. 9.22) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

9.7. **Розробка проекту з контролером для виконання циклу $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ з БІстабільним керуванням**

М е т а п р о е к т у: отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ з БІстабільним керуванням двома виконавчими механізмами.

Цикловими багато тактовими називаються системи, що виконують певну послідовність дій. При цьому основною відмінністю від *одно тактових* систем є виконання ланцюжка логічних команд в залежності від внутрішніх умов (стану системи), а не від зовнішнього впливу. Тобто дія одного пристрою зумовлюється діями іншого і т. д. Основною перевагою *багато тактових* систем є висока продуктивність і відносно мала кількість робочого персоналу, задіяного при роботі системи.

У технологічному обладнанні легкої промисловості переважно застосовуються *багато тактові цикли* роботи машин з двома виконавчими механізмами та жорсткою системою керування типу ***РВ*** (типу ***Р****озподільний* ***В****ал*) або з ручним керуванням. Сучасною альтернативою таких механізмів є комп’ютерно-інтегровані циклові механізми машин. Такі механізми мають спрощену кінематику і їх ведені ланки з робочим інструментом кінематичне не з’єднані з головним валом, а мають інформаційні (дискретно-електричні) зв’язки з одного боку з головним валом машини, а з іншого боку з портами входів і портами виходів вбудованого в технологічну машину PLC. Доцільне використання в технологічних машинах легкої промисловості і програмованих *одно*

тактових циклів $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ роботи машин з двома виконавчими механізмами.

Технічне завдання. Атоматизувати процес подачі заготовки в зону обробки та її обробку в зоні обробки. Пневмопривод 1 подає заготовку в зону обробки, після чого підводиться пневмоприводом 2 інструмент для обробки. Пневмопривод 1 відводиться, а потім відводиться пневмопривод 2. Початок роботи системи відбувається після натиснення кнопки S1, а зупинка після натиснення кнопки S2 та відпрацювання повного циклу.

Вибираємо наступне обладнання: контролер Festo FC34; два пневматичних циліндри двосторонньої дії; два БІстабільних пневматичних розподільника 5/2 з електричним керуванням (електромагніти Y1 та YN1, Y2 та YN2); кінцеві датчики переміщення штоків (XN1, X1 та XN2, X2); блок кнопок з контактами S1 та S2; чотири електромагнітних реле К1…К4; джерело живлення постійним струмом (***DC*** – ***D****irect* ***C****urrent*) 24 В.

На лабораторному стенді збираємо схему з контролером (рис. 9.26) для виконання циклу $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}.}$

Allocation List потрібно скласти з урахуванням таблиці 9.1 і рис. 9.26.

Складання команд керування для системи, яка реалізує наступний цикл:

$$\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}},$$

де 1 – пряма команда *Y1* пневмоприводу подачі;

$\overline{1}$ – зворотна команда *YN1* пневмоприводу подачі; $\overline{1}$

2– пряма команда *Y2* пневмоприводу обробки;

$\overline{2}$ – зворотна команда *YN2* пневмоприводу обробки.

Для початку роботи при натиснення кнопки *S1* повинен увімкнутись елемент пам’яті, який відповідає за початок роботи системи: $XST \Leftarrow S1$.

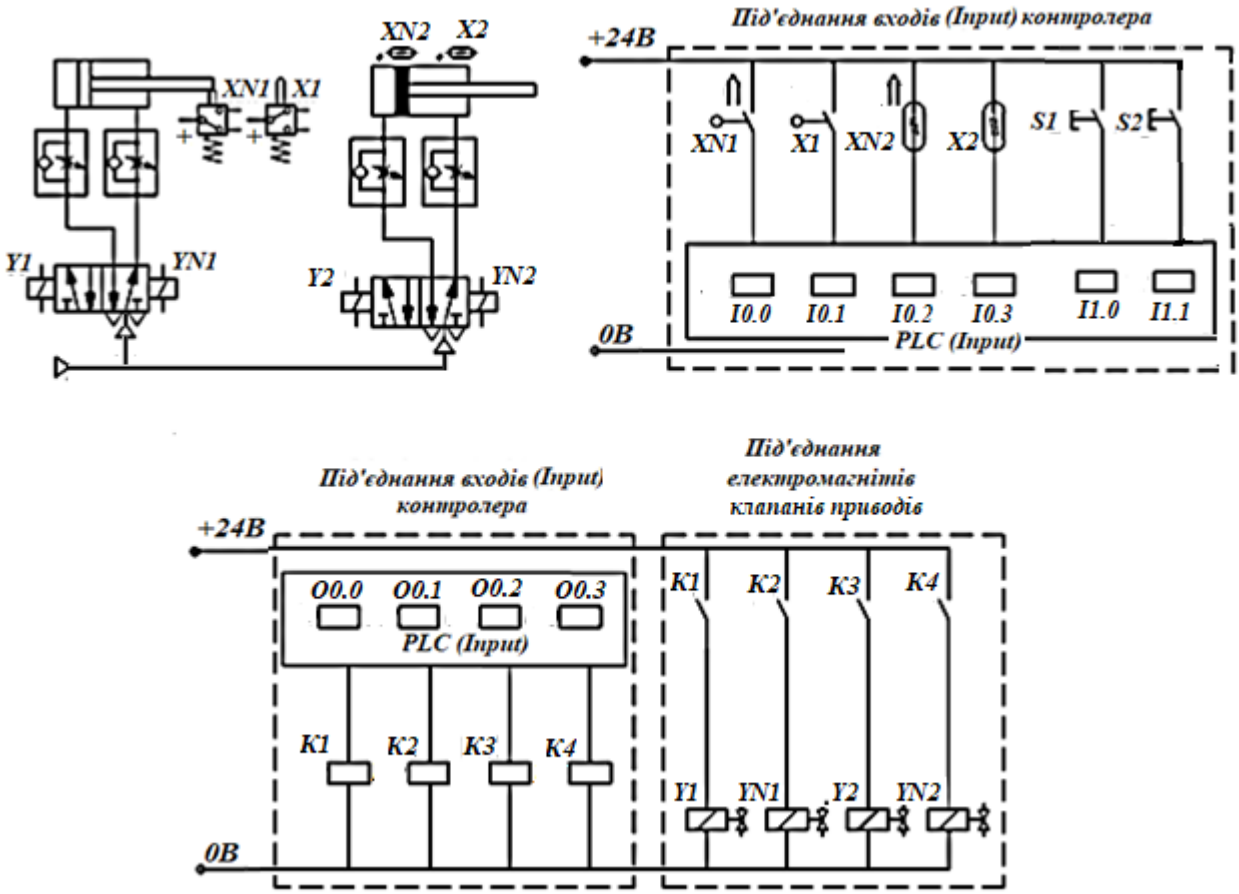


Рис. 9.26. Комбінована схема з контролером для виконання циклу $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ з БІстабільним керуванням

Команди для такого циклу будемо складати за допомогою методу графів. Для цього креслимо коло (рис. 9.27), на якому команди позначені *вершинами графу*, а лінії, які з'єднують прямі і зворотні команди для включення/вимикання відповідних електромагнітів пневмоциліндрів по логічним командам програми. При цьому вершини графу з'єднані *ребрами* (стрілками - дугами кола), які мають відповідно початок і кінець, що співпадають з відповідними вершинами графу.

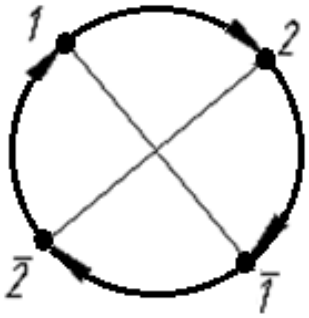
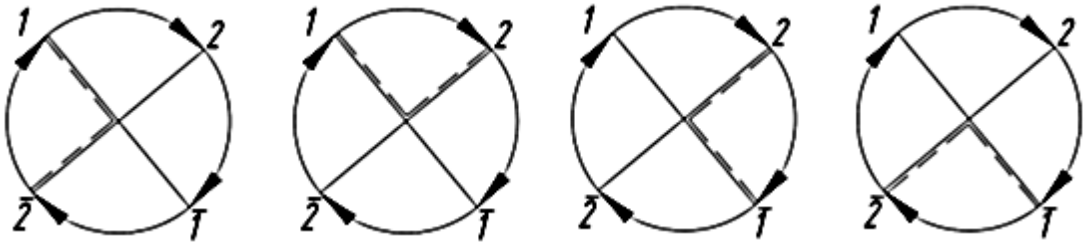


Рис. 9.27. Функціональний граф для циклу $\mathbf{1 \rightarrow 2 \rightarrow \overline{1} \rightarrow \overline{2}}$ роботи мехатронної системи з двома виконавчими механізмами і чотирма кінцевими вимикачами

Для запису команд керування необхідно зайти з кінця попередньої дуги графу відносно необхідної команди в початок необхідної дуги графу по лініям зв'язку, записуючи всі сигнали (рис. 9.28). З урахуванням наведених вимог перша (пряма) команда Y1 має наступний вигляд: $\boldsymbol{Y1 \Leftarrow XN2 \cdot XN1}$.

Як випливає з рис. 9.28,а в кінці прямої команди Y1 присутній сигнал зворотної команди XN1. XN1можна не записувати, оскільки логічне зрозуміло, що для того, щоб шток виїхав він повинен спочатку знаходитись в втягнутому положенні. Тому при написанні БІстабільної команди керування останній сигнал (останній відрізок ломаної лінії) можна не враховувати. Також в першій команді циклу $\boldsymbol{1 \to 2 \to \overline{1} \to \overline{2}}$ необхідно додати сигнал від кнопки «*Start*» XST. Тому команда Y1 для включення першого виконавчого механізму має наступний вигляд (рис. 9.28,а):

$$\boldsymbol{Y1 \Leftarrow XN2 \cdot XST} \qquad (9.25)$$



а б в г

Рис. 9.28. Фрагменти (вид) функціонального графу на рис. 9.14 для складання команд циклу $1 \to 2 \to \overline{1} \to \overline{2}$ з БІстабільним керуванням

Далі потрібно скласти наступні команди циклу за прикладом відповідно графів на рис. 9.28,б, рис. 9.28,в та рис. 9.28,г :

$$\boldsymbol{YN1 \Leftarrow X2}; \qquad (9.26)$$

$$\boldsymbol{Y2 \Leftarrow X1}; \qquad (9.27)$$

$$\boldsymbol{YN2 \Leftarrow XN1}. \qquad (9.28)$$

Розробка програми в FST4.21 по складеним логічним командам (9.25)…(9.28):

```
'Підготовка системи до пуску. Приведення
'пневмоприводів у початкове положення та вимикання
'усіх елементів пам'яті, які використовуються у
'програмі

STEP 0
 IF                NOP
 THEN  SET         YN1
       RESET       Y1
       SET         YN2
       RESET       Y2
       RESET       XST
'Перехід до STEP 1 після переведення системи в
'початковий стан

IF                    XN1
      AND             XN2
THEN         JMP TO 1

'В робочому кроці № 1 для початку роботи необхідно
'натиснути кнопку Start S1 за умови, що система
'перебуває в початковому положенні і не нетиснена
'кнопка Stop S2, яка 'відповідає за зупинку циклу.
'При виконанні цих умов вмикаємо елемент пам'яті XST,
'який був використаний при складанні команд керування

STEP 1
IF                    S1
      AND     N       S2
      AND             XN1
      AND             XN2
 THEN  SET            XST

'Далі записуємо складені команди. Важливо пам'ятати,
'що при керуванні приводами з БІстабільним керуванням
```

```
'необхідно в одній команді вмикати один сигнал і
'вимикати інший. Також бажано для зручності
'записувати команди так, як вони йдуть в циклі

IF                    XST
      AND             XN2
 THEN SET             Y1
      RESET           YN1
 IF                   X1
 THEN SET             Y2
      RESET           YN2
 IF                   X2
 THEN SET             YN1
      RESET           Y1
 IF                   XN1
 THEN SET             YN2
      RESET           Y2

'Також в програмі необхідно передбачити зупинку
'системи. В нашому випадку кнопка «Stop»  S2 вимкне
'елемент пам'яті XST і система зупиниться після
'закінчення  циклу і буде чекати повторної команди
'Start

IF                   S2
THEN  RESET          XST

'Останній крок програми необхідно завершувати
'командою на переведення системи в інший або в цей же
'крок

IF                   NOP
THEN  JMP TO 1
```

Для порівняльного аналізу на рис. 9.29 наведена комбінована схема БІстабільного керування без контролера для циклу з $\mathbf{1} \to \mathbf{2} \to \overline{\mathbf{1}} \to \overline{\mathbf{2}}$.

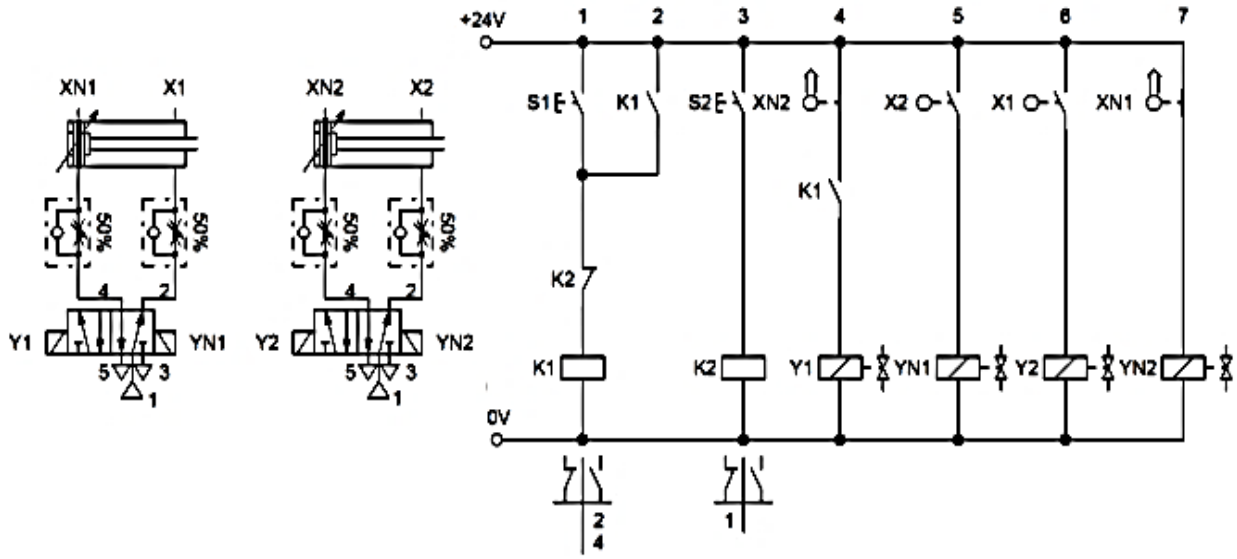


Рис. 9.29. Комбінована схема БІстабільного керування *без контролера* для виконання циклу з $\mathbf{1} \to \mathbf{2} \to \overline{\mathbf{1}} \to \overline{\mathbf{2}}$

Завдання для самостійної роботи

1. Переробити програму для керування системою для циклу $\mathbf{1} \to \mathbf{2} \to$ $\to \overline{\mathbf{1}} \to \overline{\mathbf{2}}$ для МОНОстабільного керування та одноразовим циклом роботи після натискання кнопки S1.

2. Додати кнопки S1 та S2 до схеми з контролером (рис. 9.26). Після цього порівняти комбіновану схему з комбінованою схемою без контролера (рис. 9.29) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

9.8. **Розробка проекту з контролером для циклу $\mathbf{1} \to \overline{\mathbf{1}} \to \mathbf{2} \to \overline{\mathbf{2}}$ з БІстабільним керуванням і використанням елементу пам'яті**

Мета проекту: отримання практичних навичок по програмуванню циклових систем першого класу складеності (одно режимних) на прикладі виконання циклу $\mathbf{1} \to \overline{\mathbf{1}} \to \mathbf{2} \to \overline{\mathbf{2}}$ з БІстабільним керуванням двома виконавчими механізмами і використанням елементу пам'яті.

В такій системі підчас її роботи один і той же стан повторюється двічі (на початку і після відпрацювання першого пневмоциліндру, коли обидва штоки втягнуті). В такому стані система "не знає" чи починає вона

цикл, чи вже відпрацювала половину циклу. Початок побудови функціонального графу циклу наведений на рис. 9.30.

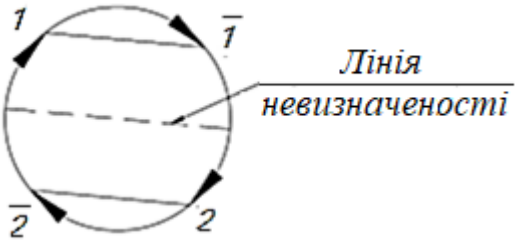


Рис. 9.30. Інформаційно невизначений граф циклу $\mathbf{1 \rightarrow \overline{1} \rightarrow 2 \rightarrow \overline{2}}$ (без елемента пам'яті)

З графу циклу видно, що лінії прямої і зворотної команд для пневмоприводу 1 та пневмоприводу 2 не перетинаються. Такий граф розпадається на два підграфи, які інформаційно не зв'язані. Ця інформаційна невизначеність позначається можливістю проведення на графі *лінії невизначеності*, яка на рис.9.30 зображена пунктирною лінією. Взагалі, лінія невизначеності проводиться так, що вона:

- не перетинає жодної лінії зв'язку прямих і зворотних команд;
- ділить граф на частини (підграфи), в кожній з яких є хоча б одна лінія зв'язку.

Якщо в графі системи є хоча б одна лінія невизначеності, то цикл керування такої системою не може бути програмно реалізованим.

Для того, щоб система стала реалізованою її необхідно доповнити додатковим функціональним модулем – *елементом пам'яті*. Елемент пам'яті *ЕР* вводиться в цикл системи таким чином, щоб в початковий момент роботи системи всі функціональні модулі системи перебували в початковому положенні. Для елемента пам'яті записуються БІстабільні команди тому, що його лінія зв'язку позначає стан включення, а саме стан «*1*» і стан вимкнення, а саме стан «*0*».

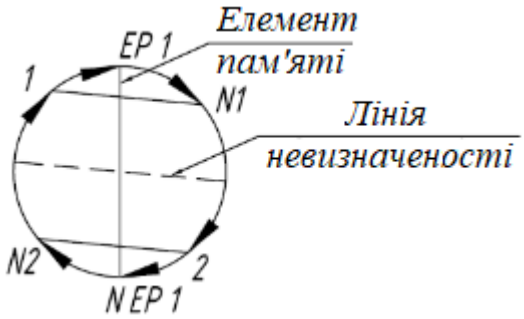


Рис. 9.31. Інформаційно визначений граф циклу
$\mathbf{1 \rightarrow EP1 \rightarrow \overline{1} \rightarrow 2 \rightarrow \overline{EP1} \rightarrow \overline{2}}$
(з елементом пам'яті *ЕР*)

Отже, в наведеному прикладі є одна лінія невизначеності, яка прибирається додатковим введенням елемента пам'яті *EP*. І тому цикл керування набуває наступного вигляду:

$$\mathbf{1} \rightarrow \boldsymbol{EP1} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{2} \rightarrow \overline{\boldsymbol{EP1}} \rightarrow \overline{\mathbf{2}} ,$$

де $\boldsymbol{EP1}$ – команда включення елементу пам'яті ***EP1***;
$\overline{\boldsymbol{EP1}} := \boldsymbol{NEP1}$ – команда вимикання елементу пам'яті ***EP1***.

Технічне завдання. Автоматизувати процес подачі заготовки в зону обробки та її обробку в зоні обробки. Привід 1 подає заготовку в зону обробки, після чого відводиться. Потім підводиться інструмент для обробки приводом 2 і потім відводиться. Початок роботи системи відбувається після натиснення кнопки S1, а зупинка після натиснення кнопки S2 після відпрацювання повного циклу.

Вибираємо наступне обладнання: контролер Festo FC34; пневматичні циліндри двосторонньої дії з БІстабільними пневматичними розподільниками 5/2 та електричним керуванням (електромагніти Y1 та YN1, Y2 та YN2); електричні кінцеві вимикачі (XN1та X1, XN2 та X2); блок кнопок з контактами S1 таS2; блок електромагнітних реле *K1…K4*; джерело живлення постійним струмом (***DC*** – ***D****irect* ***C****urrent*) 24 В.

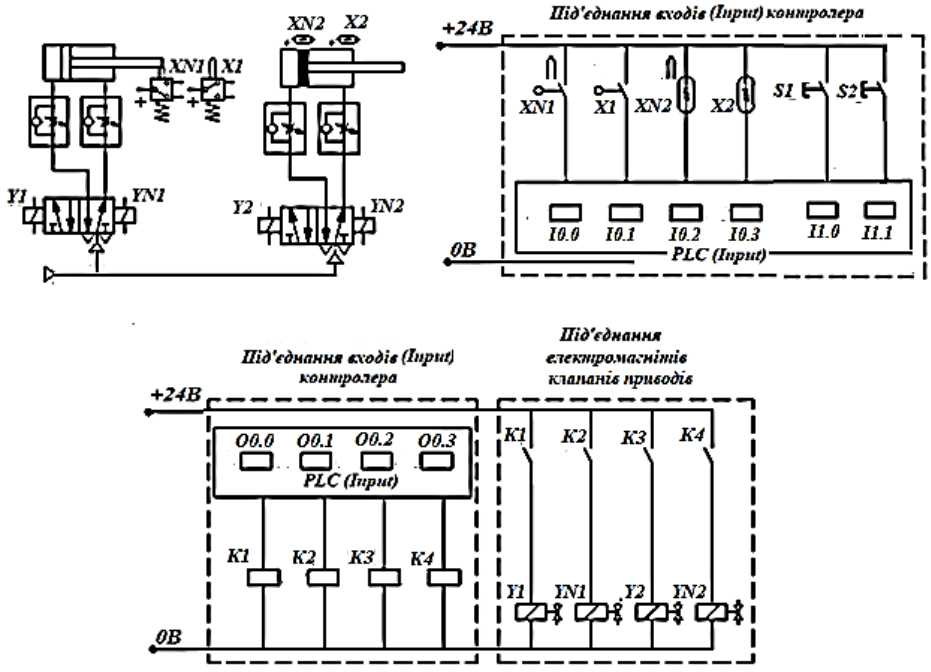


Рис. 9. 32. Схема С3 з контролером циклу $\mathbf{1} \rightarrow \boldsymbol{EP1} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{2} \rightarrow \overline{\boldsymbol{EP1}} \rightarrow \overline{\mathbf{2}}$

На рис. 9.32 наведена комбінована схема (тип С3) з контролером для виконання циклу $\mathbf{1} \rightarrow \boldsymbol{EP1} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{2} \rightarrow \overline{\boldsymbol{EP1}} \rightarrow \overline{\mathbf{2}}$. Allocation List потрібно скласти з урахуванням таблиці 9.1. На рис. 9.33 наведений приклад використання ліній зв'язку графу і рівняння причино-наслідкових зв'язків для побудові команд програми.

Створення програми в FST4.21 по складеним логічним командам (9.29)…(9.34), які наведені під графами на рис. 9.33.

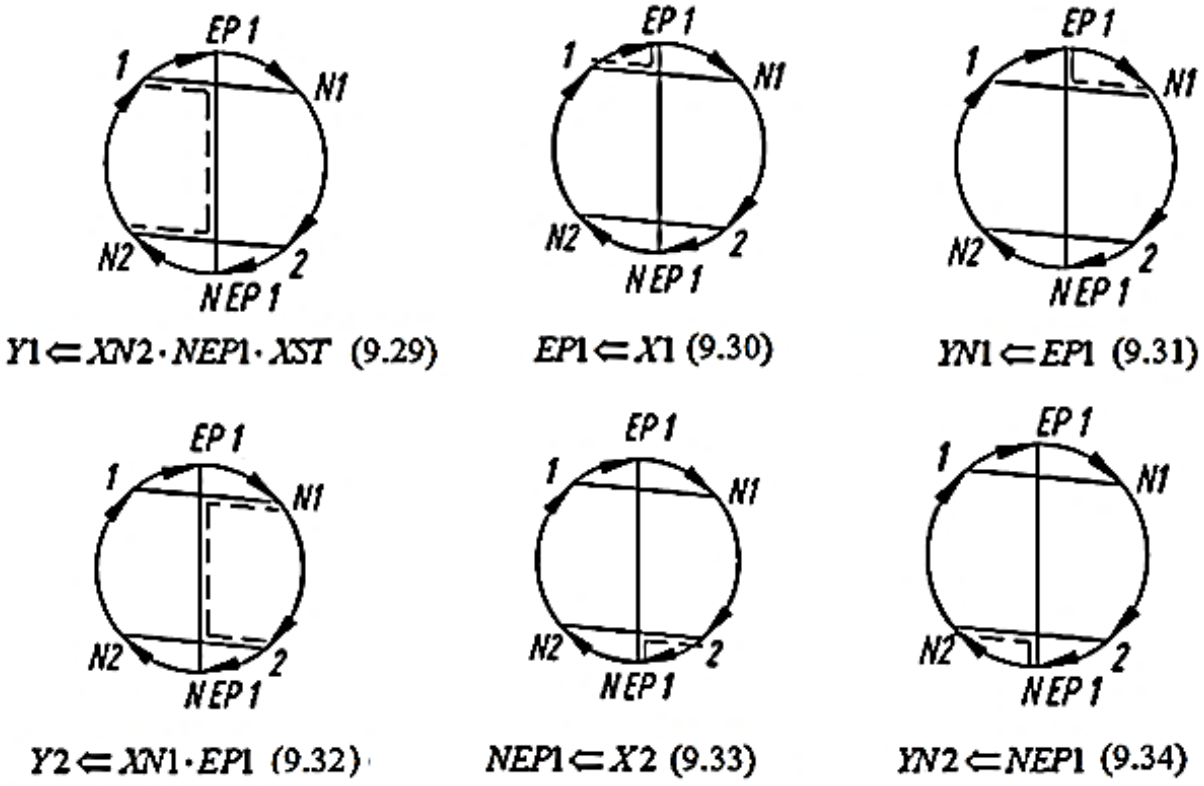


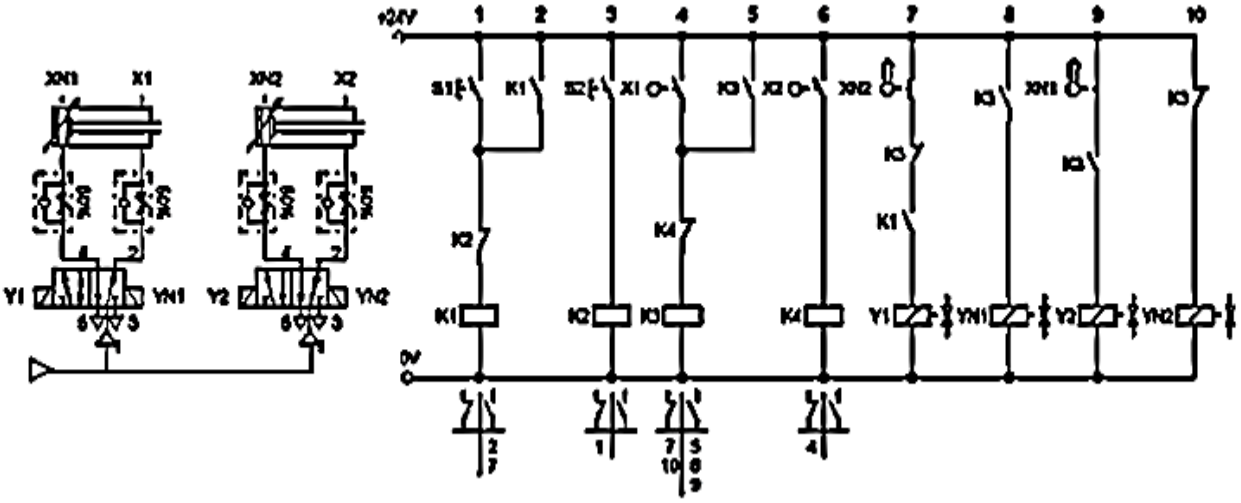
Рис. 9. 33. Фрагменти графу при використані ліній зв'язку (пунктир) для побудові логічних команд керування циклу $\mathbf{1} \rightarrow \boldsymbol{EP1} \rightarrow \overline{\mathbf{1}} \rightarrow \mathbf{2} \rightarrow \overline{\boldsymbol{EP1}} \rightarrow \overline{\mathbf{2}}$

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
'пам'яті, які використовуються в програмі

STEP 0
 IF                    NOP
 THEN  SET             YN1
```

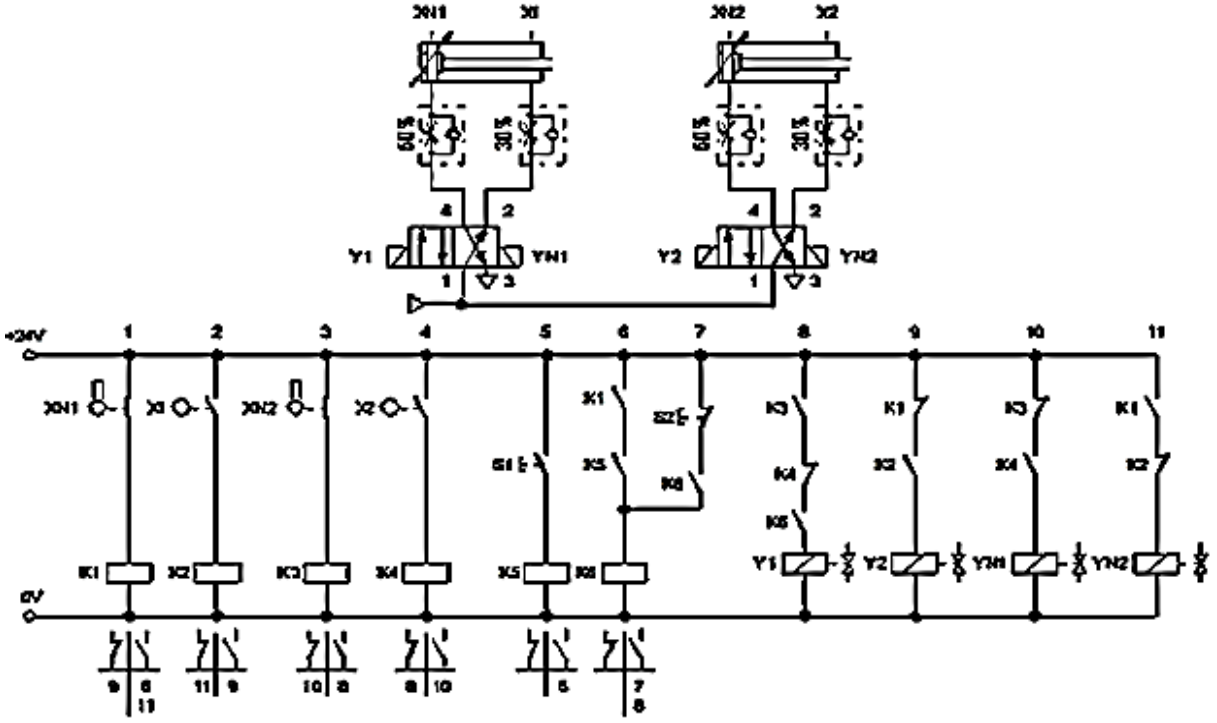
```
       RESET          Y1
       SET            YN2
       RESET          Y2
       RESET          XST
       RESET          EP1
'Перехід до STEP 1 після переведення системи в
початковий 'стан
 IF                   XN1
        AND           XN2
 THEN JMP TO 1

'В робочому кроці № 1 для початку роботи необхідно
'натиснути кнопку «Start» S1 за умови, що система
'перебуває в початковому положенні і не нетиснена
'кнопка  «Stop» S2, яка відповідає за зупинку циклу.
'При виконанні цих умов вмикаємо елемент пам'яті
'XST, який використаний при складанні команди (9.29)

STEP 1
 IF                   S1
       AND     N      S2
       AND            XN1
       AND            XN2
 THEN  SET            XST
'Далі записуємо складені команди (9.29)…(9.34).
'Важливо пам'ятати, що при керуванні приводами з
'БІстабільним керуванням необхідно в одній команді
'вмикати один сигнал і вимикати інший. Також бажано
'для зручності записувати команди так, як вони
'записані в циклі на рис.9.33

IF                    XST    'рівняння (9.29)
       AND            XN2
THEN   SET            Y1
       RESET          YN1
 IF                   X1     'рівняння (9.30)
 THEN  SET            EP1
 IF                   EP1    'рівняння (9.31)
```

```
THEN   SET              YN1
       RESET            Y1
IF                      XN1     'рівняння (9.32)
       AND              EP1
THEN   SET              Y2
       RESET            YN2
IF                      X2      'рівняння (9.33)
THEN   RESET            EP1
IF              N       EP1     'рівняння (9.34)
THEN   SET              YN2
       RESET            Y2
```

```
'Також в програмі необхідно передбачити зупинку
'циклу кнопкою S2, яка вимкне елемент пам'яті XST і
'цикл буде чекати повторної команди «Start»
    IF                      S2
    THEN   RESET            XST
'Останній крок програми необхідно завершувати
'командою на переведення системи в інший або в цей же
'крок
    IF                      NOP
    THEN   JMP              TO 1
```



а



б

Рис. 9. 34. Комбіновані схеми без контролера для виконання циклу $1 \to \overline{1} \to 2 \to \overline{2}$ **: а** – варіант 1; **б** – варіант 2

Для порівняльного аналізу на рис. 9.34 наведені комбіновані схеми без контролера для виконання циклу $1 \to \overline{1} \to 2 \to \overline{2}$, які відрізняються від функціонально адекватної схеми для такого же циклу на рис. 8.49, розділ 8.

Завдання для самостійної роботи студентів:

1. Переробити програму для керування системою з аналогічним циклом, але з двома МОНОстабільними пневматичними розподільниками та одноразовим циклом роботи після натискання кнопку «Start» S1.

2. Порівняти комбіновану схему з контролером (рис. 9.32) з комбінованими схемами без контролера (рис. 9.34) по кількості дротів і контактів електромагнітних реле, які застосовані в цих схемах.

9.9. **Розробка проекту системи другого класу з контролером для виконання циклу** $(\mathbf{1 - \overline{1}})$ x **2 без використання лічильника**

Мета проекту: отримання практичних навичок по програмуванню систем **2-го класу складності** (багато режимних) на прикладі виконання циклу $\mathbf{1 - \overline{1} - 1 - \overline{1}}$.

В автоматизованих машинах легкої промисловості можливе використання декількох програааамно керованиих виконавчих механізмів, які кінематично не з'єднані з головним валом і які виконують декілько технологічних операцій або дій.

Застосування в схемах тільки одно режимних пристроїв привело б до того, що кількість виконавчих пристроїв дорівнювало б числу операцій. А це, в свою чергу, призводить до подорожчання системи в декілька разів. Тому при створенні автоматизованих систем розробники прагнуть мінімізувати кількість пристроїв, що входять до складу системи. Одним із способів такої мінімізації є робота деяких приводів по кілька разів протягом одного циклу. Наприклад, з листа гуми чи шкіри можна вирізати дві заготовки для подальшої їх обробки. Операцію опускання лез може виконувати один і той же привід. Такий привід буде багато режимним. Багато режимними називаються пристрої, які під час одного циклу роботи системи виконують свої дії кілька разів.

Для програмування необхідного циклу потрібно записати команди керування у відповідності до функціонального графу циклу роботи механізму з наступним складанням рівнянь причино-наслідкових зв'язків.

Технічне завдання. Скласти програму керування приводом леза, яке за один робочий цикл виконує два зворотно-поступальні рухи. Переведення системи в робочий режим і вихід з робочого режиму виконується натисненням кнопки $S1$. Відпрацювання системою одного робочого циклу - після натиснення кнопки $S2$.

Вибираємо наступне обладнання: PLC Festo FC34; пневматичний циліндр двосторонньої дії з БІстабільним пневматичним розподільником 5/2 з електромагнітним керуванням (електромагніти $Y1$ та $YN1$); два пневматичні дроселі зі зворотнім клапанами; два кінцевих вимикачі $XN1$ та $X1$; кнопка «Пуск» *S1* і кнопка «Start» $S2$; блок електричних реле.

На рис. 9.35 наведена комбінована схема з контролером для виконання циклу $\mathbf{1-\overline{1}-1-\overline{1}}$

.

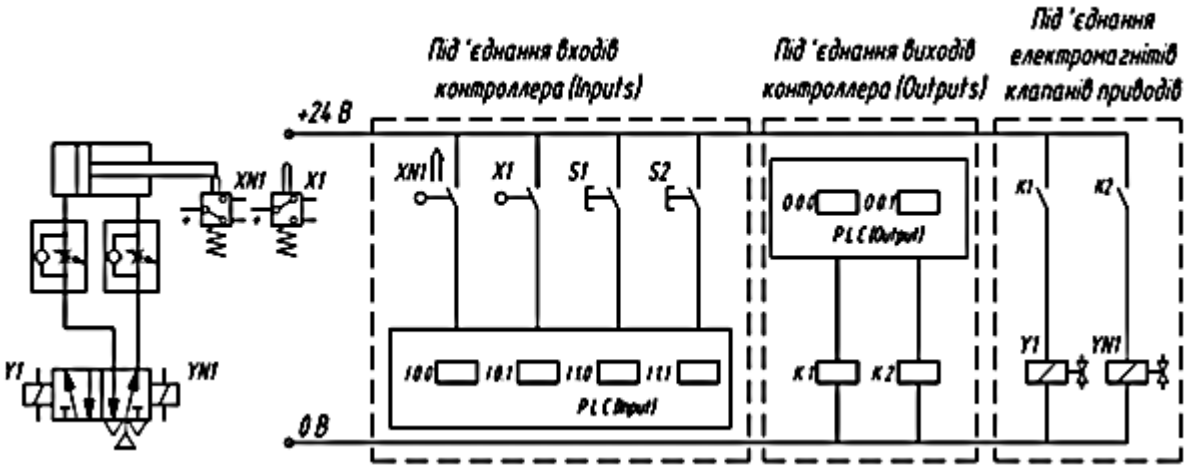


Рис. 9. 35. Комбінована схема з контролером для циклу $\mathbf{1-\overline{1}-1-\overline{1}}$

Allocation List потрібно скласти з урахуванням таблиці 9.1.

Складання логічних команд керування для багато режимних пристроїв: зобразимо граф циклу (рис. 9.36). Як видно з графа у нас є чотири лінії зв'язку, які не перетинаються.

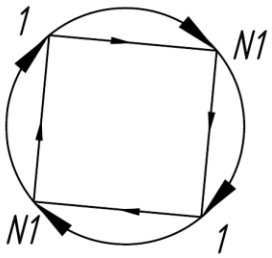


Рис. 9. 36. Граф циклу $1-\overline{1}-1-\overline{1}$

Для побудови всіх ліній невизначеності граф необхідно розбити на два підграфа. В одному підграфі будуть тільки лінії зв'язку $1 \to N1$, а в другому тільки лінії зв'язку $N1 \to 1$ (рис.9.37). Маємо лінії невизначеності в кожному під графі. Перенесемо їх на основний граф і перетинаємо двома додатковими елементами пам'яті *EP1* та *EP2* (рис. 9.38).

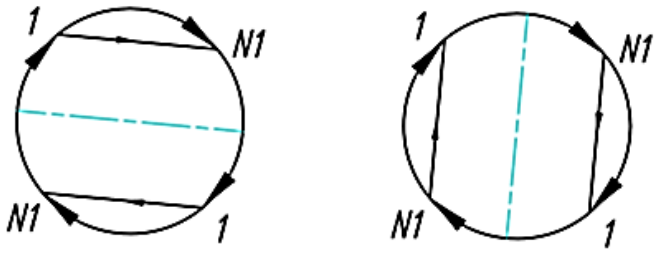


Рис. 9.37. Підграфи циклу $\mathbf{1-\overline{1}-1-\overline{1}}$

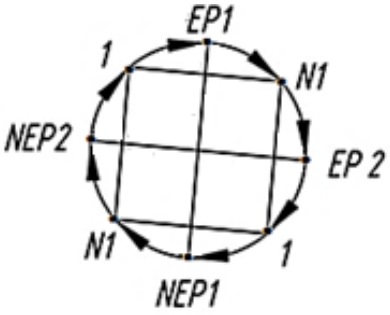


Рис. 9. 38. Граф циклу $\mathbf{1-\overline{1}-1-\overline{1}}$ з елементами пам'яті ***EP1*** і ***EP2***

На рис. 9.39 наведені функціональний граф робочого циклу і зони існування сигналів включеного (***EP1*** та ***EP2***) стану і вимкнутого (***NEP1*** та ***NEP2***) стану елементів пам'яті.

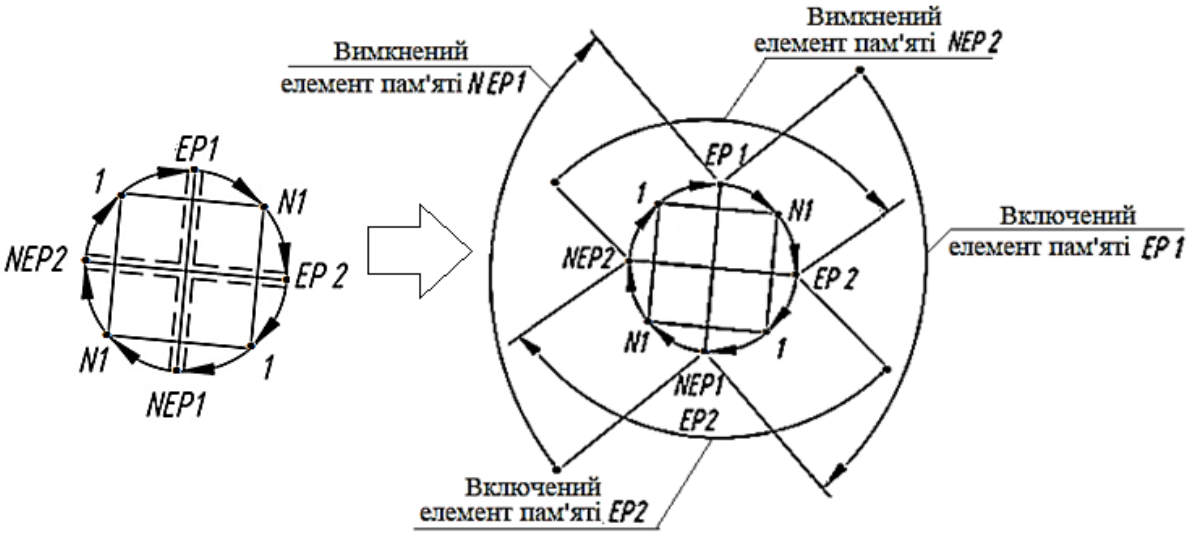


Рис. 9. 39. Граф циклу $\mathbf{1 \rightarrow EP1 \rightarrow N1 \rightarrow EP2 \rightarrow 1 \rightarrow NEP1 \rightarrow N1 \rightarrow}$ $\mathbf{\rightarrow NEP2}$ із зонами існування сигналів включеного (***EP1*** та ***EP2***) стану і вимкнутого (***NEP1*** та ***NEP2***) стану елементів пам'яті

При складанні керуючих команд необхідно уважно розділяти сигнали багато режимних пристроїв. Наприклад згідно графа на рис. 9.39 команда включення елементу пам'яті *EP1* має вигляд: $EP1=X1$.

Однак під час роботи системи за циклом сигнал *X1* проходить двічі, а система повинна відпрацювати тільки один раз в правильний момент. Для вибору правильної одиниці поглянемо на інші функціональні модулі системи і порівняємо їх стан для двох станів *X1*.

Для першого випадку при *X1 EP1=0, EP2=0*, а для другого випадку при *X1 EP1=1, EP2=1*. Отже стан системи при двох *X1* відрізняється по сигналу від елементу пам'яті *EP1* та *EP2,* але оскільки ми записуємо команду для *EP2*, от використовуємо сигнал *EP2=0,* тому команда на вмикання $EP1 \Leftarrow X1 \cdot NEP2$.

Для багато режимних пристрої будуть мати дві команди на вмикання і вимикання, які будуть об'єднанні знаком логічного додавання «+». Також при складанні БІстабільних команд для багато режимних пристрої необхідно обмежувати керуючі сигнали так, щоб вони не були рівні одиниці під час відпрацювання зворотних команд. В команду *Y1* додаємо сигнал від кнопки *S2* за умовою технічного завдання:

$$\boldsymbol{Y1 \Leftarrow NEP2 \cdot S2 \cdot NEP1 + EP2 \cdot EP1.} \quad (9.35)$$

Інші команди мають наступний вигляд:

$$\boldsymbol{EP1 \Leftarrow X1 \cdot NEP2\ ;} \quad (9.36)$$

$$\boldsymbol{YN1 \Leftarrow EP1 \cdot NEP2 + NEP1 \cdot EP2\ ;} \quad (9.37)$$

$$\boldsymbol{EP2 \Leftarrow XN1 \cdot EP1;} \quad (9.38)$$

$$\boldsymbol{NEP1 \Leftarrow X1 \cdot EP2;} \quad (9.39)$$

$$\boldsymbol{NEP2 \Leftarrow XN1 \cdot NEP.} \quad (9.40)$$

Згідно технічного завдання на створення проекту переведення системи в робочий режим виконується при першому натисканні кнопки *S1* і вихід з робочого режиму виконується при другому натисканні цієї же кнопки *S1*. Тобто сигнали, які буде сприймати система - це дворазове натиснення та відтиснення кнопки *S1* для наступного додаткового циклу натискання кнопки:

$$\boldsymbol{S1 \rightarrow NS1 \rightarrow S1 \rightarrow NS1.} \quad (9.41)$$

На рис. 9.40 наведені функціональний граф циклу і зони обов'язкового існування сигналів включеного стану натиснення кнопки (*XST*) і елемента пам'яті (*EP3*) та вимкнутого стану натиснення кнопки

(*NXST*) та (*NEP3*) додаткових елементів пам'яті, які перетворюють цикл (9.41) у цикл (9.42):

$$\boldsymbol{S1 \rightarrow XST \rightarrow NS1 \rightarrow EP3 \rightarrow S1 \rightarrow NXST \rightarrow NS1 \rightarrow NEP3} \qquad (9.42)$$

$$\boldsymbol{XST \Leftarrow S1 \cdot NEP3\ ;} \qquad (9.43)$$

$$\boldsymbol{EP3 \Leftarrow NS1 \cdot XST\ ;} \qquad (9.44)$$

$$\boldsymbol{NXST \Leftarrow S1 \cdot EP3\ ;} \qquad (9.45)$$

$$\boldsymbol{NEP3 \Leftarrow NS1 \cdot NXST\ .} \qquad (9.46)$$

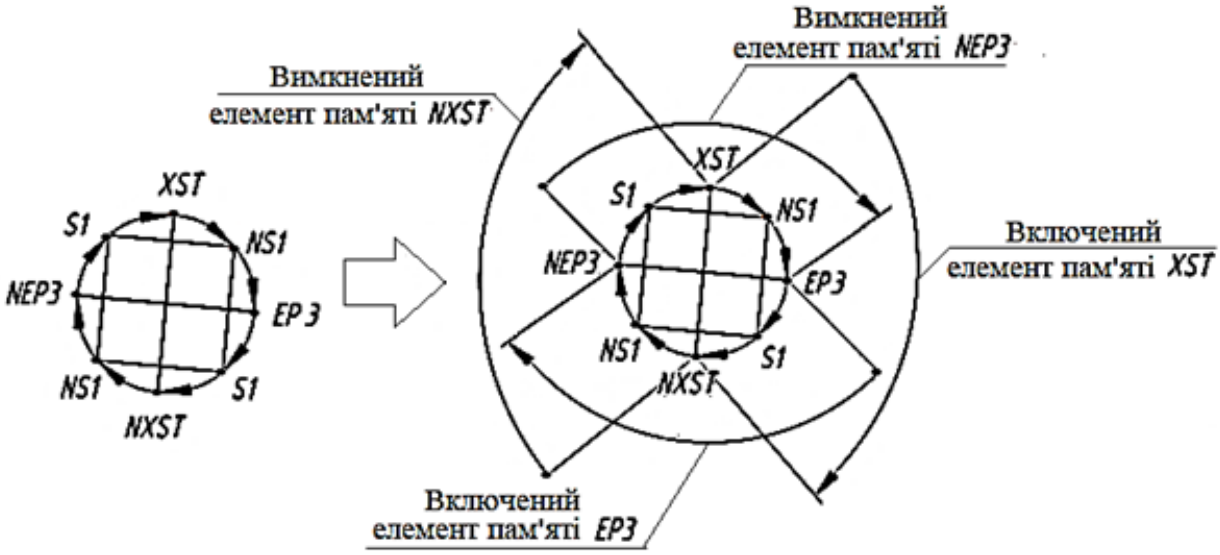


Рис. 9.40. Функціональний граф циклу натиснення кнопки (а) і зон обов'язкового існування сигналів включеного (***XST*** та ***EP3***) і вимкнутого (***NXST*** та ***NEP3***) стану додаткових елементів пам'яті

Оскільки елемент пам'яті ***XST*** ереводить систему в робочий режим то його необхідно додати в першу частину команди ***Y1*** і вираз (9.35) отримує наступний вигляд:

$$\boldsymbol{Y1 \Leftarrow NEP2 \cdot S2 \cdot NEP1 \cdot XST + EP2 \cdot EP1\ .} \qquad (9.47)$$

Створення програми в FST4.21 по складеним логічним командам:

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
'пам'яті, які використовуються в програмі

STEP 0
 IF                    NOP
 THEN  SET             YN1
       RESET           Y1
       RESET           XST
       RESET           EP1
       RESET           EP2
       RESET           EP3

'Для переходу в робочий крок необхідно щоб поршень
'циліндру був втягнутими
IF                   XN1
THEN  JMP TO 1

'В робочому кроці записуємо раніше складені команди.
'На початку записуємо команди, пов'язані з
'натисненням кнопки S1
STEP 1
 IF                    S1
       AND     N       EP3
 THEN  SET             XST          '(9.43)
 IF            N       S1
       AND             XST
 THEN  SET             EP3          '(9.44)
 IF                    S1
       AND             EP3
 THEN  RESET           XST          '(9.45)
 IF            N       S1
       AND     N       XST          '(9.46)
 THEN  RESET           EP3

'Далі записуємо команди робочого циклу
```

```
IF              N       EP2
       AND              S2
       AND      N       EP1
       AND              XST
 THEN  SET              Y1  'перший доданок виразу(9.47)
       RESET            YN1
 IF                     X1
       AND      N       EP2
 THEN  SET              EP1     '(9.36)
 IF                     EP1
       AND      N       EP2
 THEN  SET              YN1'перший доданок виразу(9.37)
       RESET            Y1
 IF                     XN1
       AND              EP1
 THEN  SET              EP2     '(9.38)
 IF                     EP2
       AND              EP1
 THEN  SET              Y1  'другий доданок виразу(9.47)
       RESET            YN1
 IF                     X1
       AND              EP2
 THEN  RESET            EP1     '(9.39)
 IF              N      EP1
       AND              EP2
 THEN  SET              YN1'другий доданок виразу(9.37)
RESET           Y1
 IF                     XN1
       AND      N       EP1
 THEN  RESET            EP2     '(9.40)

'Останній крок програми необхідно завершувати
'командою на переведення системи в інший або в цей
'же крок

IF                    NOP
THEN  JMP TO 1
```

Завдання для самостійної роботи студентів. Переробити програму для керування системою з аналогічним циклом, але з двома МОНОстабільними пневморозподільниками. Переводити систему в робочий режим після відтиснення попередньо нетисненої кнопки S1.

9.10. **Розробка проекту контролером для виконання циклу $(\mathbf{1-\overline{1}})$ x $\mathbf{5}$ з використанням лічильника**

Мета проекту: отримання практичних навичок по програмуванню систем **2-го класу складеності** (багато режимних) на прикладі виконання циклу $(1-\overline{1})$ x 5 з використанням лічильника, який задається програмно.

При створенні автоматизованих системах легкої промисловості виникають задачі забезпечення повторення певної операції багато раз. Для вирішення таких задач можна використати графи циклу з багато режимними пристроями, однак в такому випадку збільшується кількість елементів пам'яті та ускладняється запис логічних команд керування для них.

В усіх сучасних контролерах є вбудовані операнди-лічильники, які рахують кількість імпульсів, що поступають їм на вхід.

При створенні програму роботи системи з лічильником важливо зробити такі умови, щоб лічильник вмикався тільки необхідну кількість раз під час циклу і рахував імпульси, які на нього поступають. Тому при створенні програму будуть використанні додаткові елементи пам'яті.

Для програмування необхідного циклу потрібно записати команди керування у відповідності до функціонального графу циклу роботи механізму з наступним складанням рівнянь причино-наслідкових зв'язків.

Технічне завдання. Розробити програму БІстабільного керування приводом для вивантаження заготовок в зону обробки. У магазинному завантажувальному пристрої знаходиться п'ять заготовок. Після натиснення кнопки S1 ці п'ять заготовок відвантажуються. При повторному натисненні кнопки S1 цикл повторюється. При натисненні кнопки S2 привід повертається в початкове положення і система чекає натиснення кнопки S1 для включення циклу.

Вибираємо наступне обладнання: контролер Festo FC34; пневматичний циліндр двосторонньої дії; БІстабільний

пневморозподільник 5/2 з електромагнітним керуванням (електромагніти *Y1* та *YN1*); пневматичні дроселі зі зворотними клапанами; кінцеві вимикачі *XN1* та *X1*; кнопка «Start» *S1* і кнопка «Start» *S2*; блок електромагнітних реле *K1* і *K2*; джерело живлення постійним струмом (***DC*** – ***D****irect* ***C****urrent*) 24 В.

На рис. 9.41 наведена комбінована схема з контролером для виконання з циклу $(\mathbf{1}-\overline{\mathbf{1}})$ x $\mathbf{5}$ з лічильником С0 (*Counter*), який задається програмно і тому на схемі відсутній.

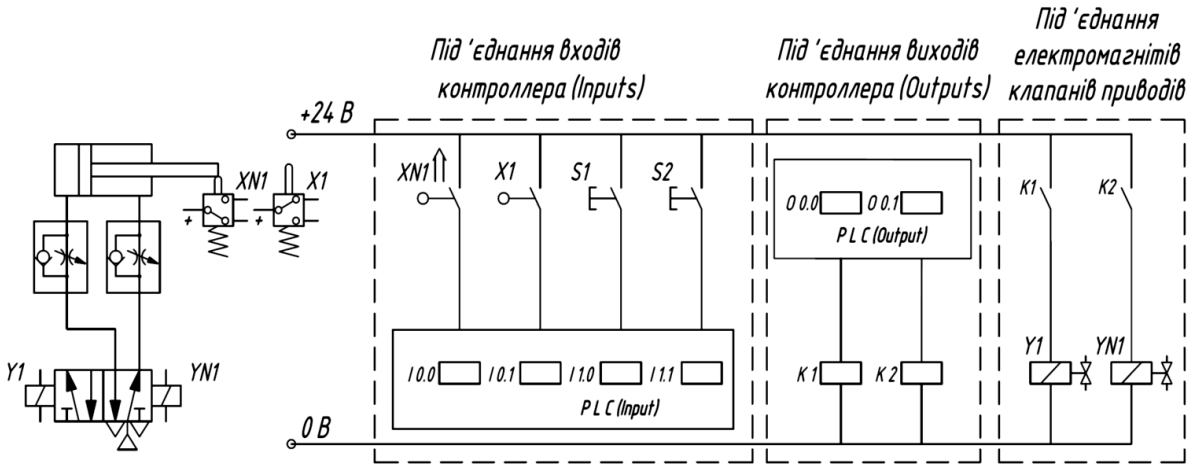


Рис. 9.41. Комбінована схема з контролером для виконання циклу $(\mathbf{1}-\overline{\mathbf{1}})$ x $\mathbf{5}$

Allocation List потрібно скласти з урахуванням таблиці 9.1.

Складання логічних команд керування.

В мові STL включення лічильника С0 відбувається наступним чином:

```
IF <умова> AND C0
THEN LOAD V5
TO CW0
SET C0,
```

де **C0** – лічильник повторення циклів $(1-\overline{1})$;

CW0 – преселектор лічильника ;

THEN LOAD V5 – команда «ТОДІ ЗАВАНТАЖИТИ 5 циклів (**V5**) для лічильника С0» .

Зарахування однієї одиниці преселектору лічильника при лічбі виконується наступним чином:

IF <умова>
AND ***N EP1***
THEN INC CW0
SET ***EP1,***

де ***EP1*** – елемент пам'яті, який забезпечує одноразове додавання одиниці до преселектора лічильника,

INC CW0 – команда «ЗБІЛЬШЕННЯ на одиницю преселектора лічильника (**CW0**)», яке відбувається при умові вимкненого елементи пам'яті ***EP1***.

При створенні програми для циклу з лічильником циклів можна розглядати два графи - граф загального циклу (рис. 42,а) і підграф циклу лічильника (рис. 42,б).

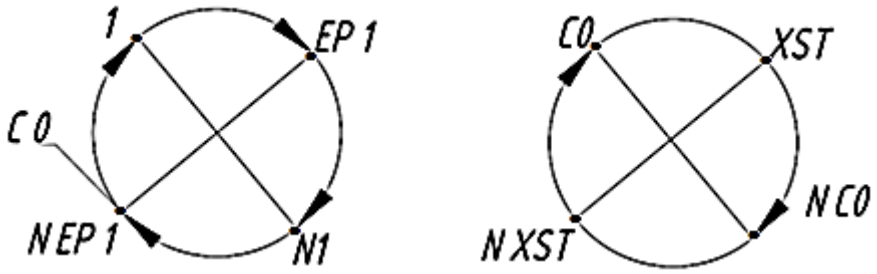


Рис. 9.42. Графи циклу: а – граф загальний циклу; б – підграф циклу лічильника

Спочатку потрібно скласти команди для підграфу циклу лічильника:

$$\mathbf{C0} \Leftarrow \boldsymbol{S1 \cdot NS2 \cdot XN1 \cdot NC0 \cdot NXST};\qquad (9.48)$$

Команда на включення елемента пам'яті ***XST*** (рис. 9.38,б) буде аналогічна команді **C0** (9.52):

$$\boldsymbol{XST} \Leftarrow \boldsymbol{S1 \cdot NS2 \cdot XN1 \cdot NC0 \cdot NXST};\qquad (9.49)$$

Але на відміну від команди для лічильника **C0**, який при відрахуванні заданої кількості імпульсів відмикається сам, елемент пам'яті ***XST*** необхідно вимкнути командою (9.54).

Записуємо логічні команди керування для графу загального циклу:

$$Y1 \Leftarrow C0 \cdot NEP1\,; \tag{9.50}$$

$$EP1 \Leftarrow X1 \cdot NEP1\,; \tag{9.51}$$

$$YN1 \Leftarrow EP1\,; \tag{9.52}$$

$$NEP1 \Leftarrow XN1 + S2\,; \tag{9.53}$$

$$NXST \Leftarrow S2 + NC0 \cdot XST\,; \tag{9.54}$$

$$NC0 \Leftarrow S2\,. \tag{9.55}$$

Оскільки вмикання і вимикання елементів пам'яті контролером виконується миттєво то можна об'єднати команди ***Y1*** та ***NEP1*** і ***YN1*** та ***EP1***.

Створення програми в FST4.21 по складеним логічним командам:

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
'пам'яті, які використовуються в програмі

STEP 0
 IF                  NOP
 THEN  RESET         Y1
       SET           YN1
       RESET         EP1
       RESET         C0
       RESET         XST

'Для переходу в робочий крок необхідно щоб циліндр
'був втягнутим
IF                 XN1
THEN  JMP TO 1

'Далі записуємо програму згідно складених команд
```

```
STEP 1
 IF                    S1
       AND     N       S2
       AND             XN1
       AND     N       C0
       AND     N       XST
 THEN  LOAD            V5
         TO            CW0
       SET             C0        '(9.48)
       SET             XST       '(9.49)
 IF                    C0
       AND             XN1
 THEN  SET             Y1        '(9.50)- включити Y1
       RESET           YN1
       RESET           EP1
 IF                    X1
       AND     N       EP1
 THEN  SET             EP1        '(9.51)
       INC             CW0
       SET             YN1        '(9.52)- включити YN1
       RESET           Y1
 IF            N       C0
       AND             XST
 THEN  RESET           XST        '(9.54)
 IF                    S2
 THEN  RESET           EP1        '(9.53)
       RESET           XST        '(9.54)
       RESET           C0         '(9.55)


'Останній крок програми необхідно завершувати
командою на 'переведення системи в інший або в цей же
крок

IF                    NOP
THEN  JMP TO 1
```

Завдання для самостійної роботи. Розробити програму керування системою відвантаження заготовок з МЗП після завантаження заготовок. Імітація завантаження заготовок в МЗП по натисненні кнопки $S2$. Відвантаження заготовок виконується після натиснення кнопки $S1$ і виконується пневмоциліндром з МОНОстабільним керування.

9.11. **Розробка проекту для виконання циклу $\mathbf{1-\overline{1},2-3-\overline{2},\overline{3}}$ з одночасною роботою декількох приводів**

Для скорочення часу складання виробів чи обробки деталей, а відповідно, збільшення продуктивності, окремі операції можна виконувати одночасно. Системи із застосуванням одночасно працюючих пристроїв називаються асинхронними. Відразу виникає питання, чому ж одночасні дії в цьому випадку не будуть синхронним ? По-перше, кілька приводів, які отримали одночасно керуючу команду, виконують різні технологічні функції, і тривалість їх виконання визначається технологією, а не одночасною подачею команд. По-друге, навіть для однакових функцій, наприклад, подача комплектуючих на складальну позицію, час їх виконання буде залежати від ваги деталі, стану обладнання, тиску в лініях системи, довжини сполучних трубок і багатьох інших факторів.

Мета проекту: отримання практичних навичок по програмуванню системи **3-го класу складеності** (з одночасною роботою декількох приводів), яка працює за циклом $\mathbf{1-\overline{1},2-3-\overline{2},\overline{3}.}$

Технічне завдання. Розробити програму керування для системи обробки заготовки. Після натиснення кнопки S1 приводом 1 заготовка подається в зону обробки, після чого відбувається одночасне затиснення заготовки приводом 2 і відведення приводу подачі заготовки 1 в початкове положення. Після завершення попередніх операцій відбувається обробка заготовки приводом 3. По завершенні обробки привід затиснення 2 та привід обробки 3 повертаються в початкове положення. При повторному натисненні S1 цикл повторюється.

На рис. 9.43 наведена комбінована схема з контролером для виконання циклу $\mathbf{1-\overline{1},2-3-\overline{2},\overline{3}.}$

В схемі задіяно наступне обладнання: три пневматичні циліндри двосторонньої дії; два БІстабільних пневматичного розподільника5/2 з електричним керуванням (електромагніти *Y1* та *YN1* і електромагніти *Y2* та *YN2*; один МОНОстабільний пневморозподільник 5/2 з електричним керуванням (електромагніти *Y3* та *YN3*); контактні датчики кінцевого положення (*XN1*, *X1* та *XN3*, *X3*); безконтактні датчики кінцевого положення - геркони (*XN2*, *X2*); блок кнопок *S1*, *S2*; п‘ять електромагнітних реле *K1*…*K5*; блок DC живлення 24 В.

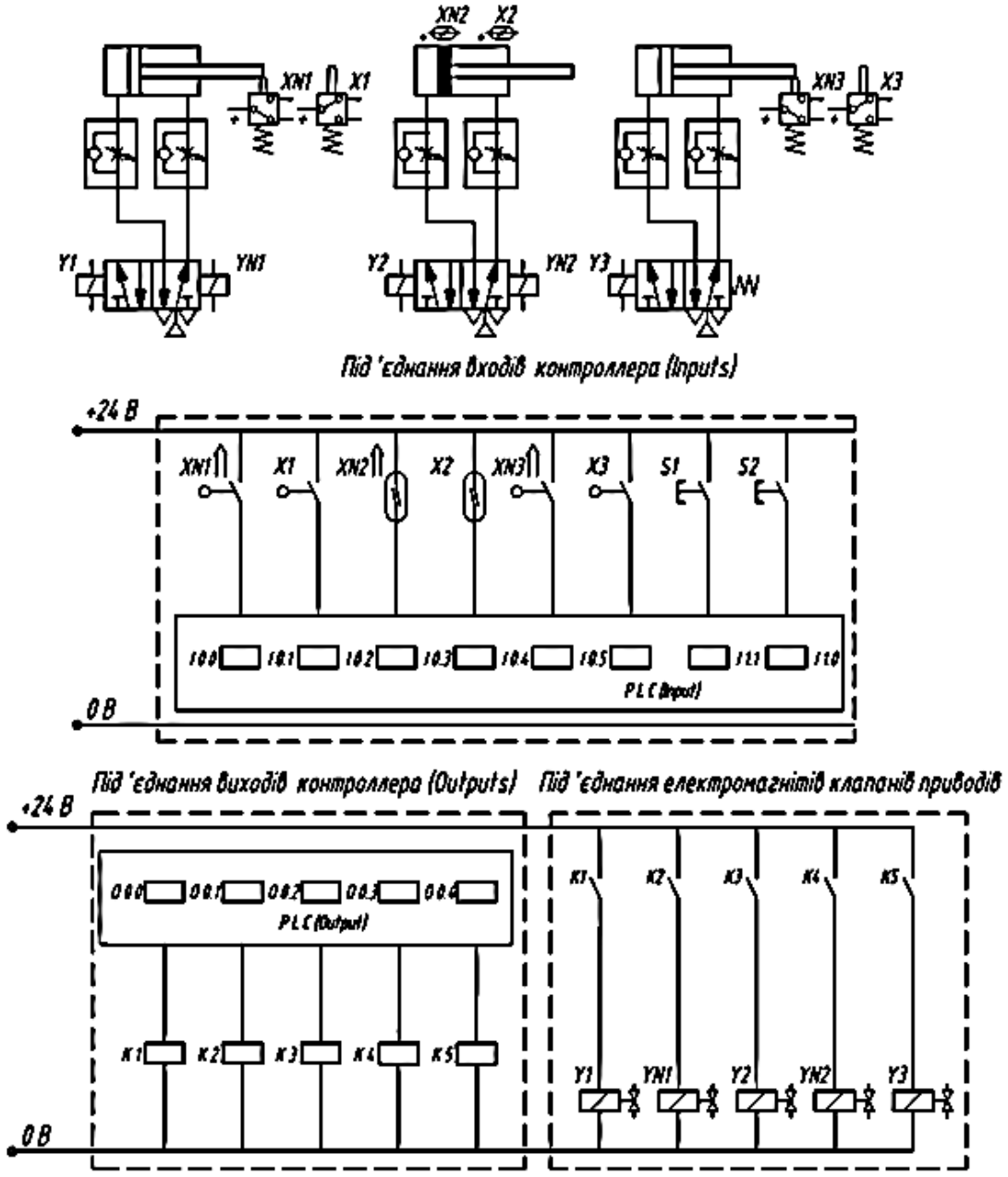


Рис. 9.43. Схема С3 з контролером для виконання циклу

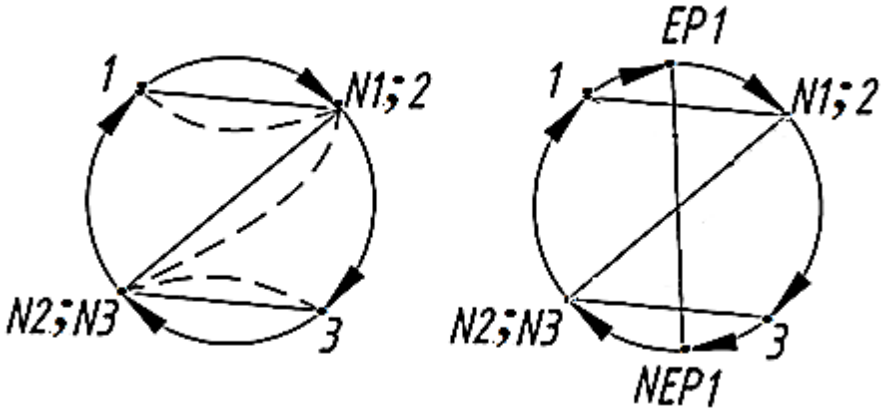
$\mathbf{1 - \overline{1}, 2 - 3 - \overline{2}, \overline{3}}$

Allocation List потрібно скласти з урахуванням таблиці 9.1.

Складання логічних команд керування: потрібно скласти граф циклу (рис. 9.44, а). Як видно, в графі є три лінії невизначеності (пунктирні), які прибираються одним елементом пам'яті *EP1* (рис. 9.44,б). В результаті функціональний граф циклу приймає наступний вигляд:

$$\mathbf{1 - EP1 - \overline{1}, 2 - 3 - NEP1 - \overline{2}, \overline{3}}$$

Особливістю складання керуючих команд для систем 3-го класу складності є те, що коли з попередньої вершини графу виходять декілька ліній зв'язку, то їх треба всі врахувати. А також команди будуть однакові, якщо їх лінії зв'язку входять в одну вершину.



а б

Рис. 9.44. Аналіз графу для виконання циклу $\mathbf{1 - \overline{1}, 2 - 3 - \overline{2}, \overline{3}}$:
а – граф з пунктирними лініями інформаційної невизначеності;
б – з елементом пам'яті $\boldsymbol{EP}$, який поглинає невизначеність графу

Рівняння причино-наслідкових зв'язків складаємо для вершин графу за стрілкою годинника, починаючі з вершини 1 графу. Записуємо систему рівнянь причино-наслідкових зв'язків (логічних команд) у відповідності до фрагментів (рис. 9.45) графу при БІстабільному керуванні виконавчими механізмами 1 і 2 та МОНОстабільному керуванні третім виконавчим механізмом, наведених на рис. 9.43:

$$\boldsymbol{Y1 \Leftarrow XN3 \cdot XN2 \cdot NEP1} \qquad \text{(рис. 9.45,а);} \qquad (9.56)$$

$$\boldsymbol{EP1 \Leftarrow X1} \qquad \text{(рис. 9.45,б);} \qquad (9.57)$$

$$\boldsymbol{YN1, Y2 \Leftarrow EP1} \qquad \text{(рис. 9.45,в);} \qquad (9.58)$$

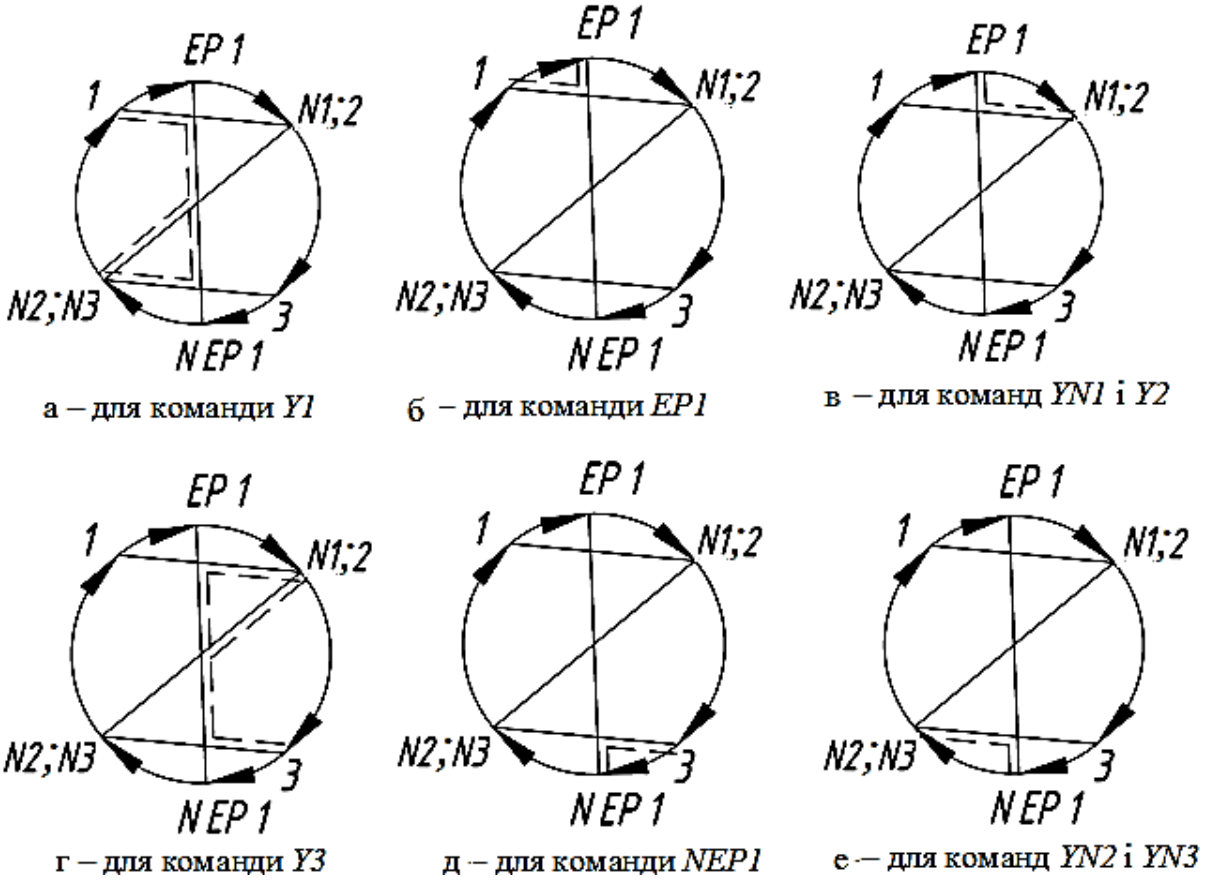
$$\boldsymbol{Y3 \Leftarrow XN1 \cdot X2 \cdot EP1} \qquad \text{(рис. 9.45,г);} \qquad (9.59)$$

$$\boldsymbol{YN2, YN3 \Leftarrow NEP1} \qquad \text{(рис. 9.45,д);} \qquad (9.60)$$

Оскільки робота системи відбувається тільки після натиснення кнопки *S1*, то команда (9.56) циклу має наступний вигляд:

$$\boldsymbol{Y1 \Leftarrow XN3 \cdot XN2 \cdot NEP1 \cdot S1} \qquad (9.61)$$

Порядок аналізу графу для складання логічних команд керування (9.56)…(9.61) наведений на рис. 9.45.



а – для команди *Y1* б – для команди *EP1* в – для команд *YN1* і *Y2*

г – для команди *Y3* д – для команди *NEP1* е – для команд *YN2* і *YN3*

Рис. 9.45. Фрагменти графа для складання логічних команд керування

Створення програми в FST4.21 по складеним логічним командам:

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
'пам'яті, які використовуються в програмі

STEP 0
 IF                    NOP
 THEN  SET             YN1
       RESET           Y1
       SET             YN2
       RESET           Y2
       RESET           Y3
       RESET           EP1
'Для переходу в робочий крок необхідно щоб циліндри
'були втягнутими

 IF                    XN1
       AND             XN2
       AND             XN3
 THEN  JMP TO 1
'Далі записуємо програму згідно складених логічних
команд 'керування

STEP 1
 IF                    S1
       AND             XN3
       AND             XN2
       AND     N       EP1
 THEN  SET             Y1
       RESET           YN1
 IF                    X1
 THEN  SET             EP1
 IF                    EP1
 THEN  SET             YN1
       RESET           Y1
       SET             Y2
       RESET           YN2
```

```
 IF                    XN1
       AND             X2
       AND             EP1
 THEN  SET             Y3
 IF                    X3
 THEN  RESET           EP1
 IF              N     EP1
 THEN  SET             YN2
 RESET                 Y2
 RESET                 Y3
'Останній крок програми необхідно завершувати командою на 'переведення системи в інший або в цей же крок
IF     NOP
THEN   JMP TO 1
```

Завдання для самостійної роботи. Розробити програму керування з трьома пневмоприводами, але які будуть працювати по зміненому циклу $\mathbf{1,2,3-\overline{1}-\overline{2}\ -\overline{3}}$. Початок роботи циклової системи: після натиснення кнопки *S1*, цикл повторюється до моменту натиснення кнопки S2. У випадку натиснення кнопки *S2* система допрацьовує останній цикл і очікує нового натиснення кнопки *S1*.

9.12. **Розробка проекту з альтернативними циклами** $\mathbf{1,2-3-4-\overline{1}-\overline{2}\ -\overline{3}-\overline{4}}$ та $\mathbf{1-2-3-\overline{2}-\overline{3}\ -2-\ \overline{1}-4-\overline{2}-\overline{4}}$

Проектування технологічного обладнання, яке виконує тільки одну технологічну операцію найчастіше є економічно невигідним. Тому часто при проектуванні подібного обладнання передбачають можливість вибору між декількома операціями, які можливо виконати на даному обладнанні. Також деякі розробники передбачають можливість додаткового програмування PLC технологічного обладнання для створення системи керування для виконання додаткових технологічних операцій.

До такого технологічного обладнання можна віднести різноманітні маніпулятори, складальні та фасувальні лінії, автоматичне та напівавтоматичне пресове та штампувальне обладнання.

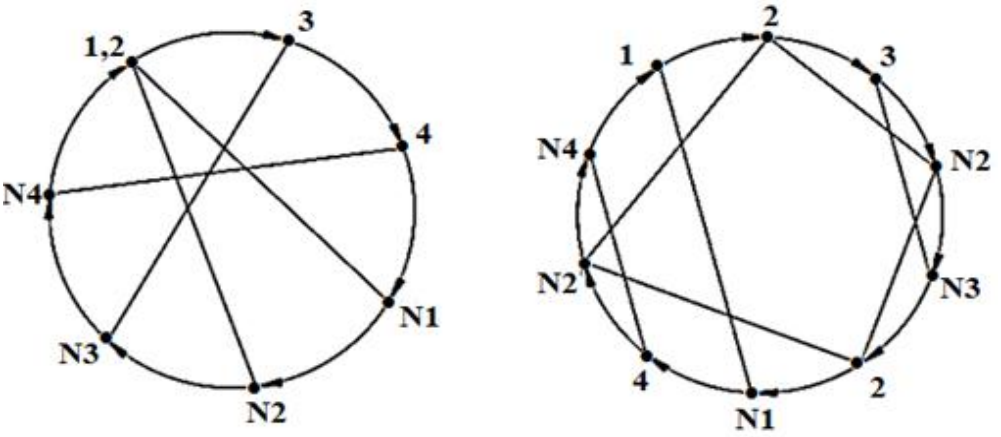
Мета проекту: отримання практичних навичок при створенні циклових систем для двох альтернативних циклів $\mathbf{1, 2-3-4-\overline{1}-\overline{2}-\overline{3}-\overline{4}}$ та $\mathbf{1-2-3-\overline{2}-\overline{3}-2-\overline{1}-4-\overline{2}-\overline{4}}$

Технічне завдання. Розробити програму керування технологічним обладнанням з чотирма приводами, яке може виконувати технологічні операції для двох альтернативних циклів $\mathbf{1, 2-3-4-\overline{1}-\overline{2}-\overline{3}-\overline{4}}$ та $\mathbf{1-2-3-\overline{2}-\overline{3}-2-\overline{1}-4-\overline{2}-\overline{4}.}$

Початок роботи системи можливий після появи сигналу від кнопки з фіксатором «Пуск» S3. За початок першого циклу відповідає кнопка «Start 1» S1, а за початок другого – «Start» S2. Перехід між циклами – тільки при відпрацюванні попереднього циклу чи за умови, що всі пневмоприводи системи знаходяться в початковому положенні. При зникненні сигналу від кнопки «Пуск» S3 – подальша робота системи неможлива, до повторної появи сигналу «Пуск» S3.

Вибираємо наступне обладнання: механізми приводів 1 та 2 - пневматичні циліндри двосторонньої дії з БІстабільними пневматичними розподільниками 5/2 з електричним керуванням (електромагніти $\boldsymbol{Y1}$ та $\boldsymbol{YN1}$ і електромагніти $\boldsymbol{Y2}$ та $\boldsymbol{YN2}$); механізми приводів 3 та 4 - пневматичні циліндрі двосторонньої дії з МОНОстабільними пневматичними розподільниками 5/2 з електричним керуванням (електромагніти $\boldsymbol{XN1}$, $\boldsymbol{X1}$, $\boldsymbol{XN2}$, $\boldsymbol{X2}$, $\boldsymbol{XN3}$, $\boldsymbol{X3}$, $\boldsymbol{XN4}$, $\boldsymbol{X4}$); блок кнопок з контактами S1, S2, S3; блок електромагнітних реле.

Для складання логічних команд керування: будуємо графи циклів (рис. 9.46). По графам циклів потрібно скласти логічні команди керування (виконати самостійно).



а б

Рис. 9.46.Функціональні графи альтернативних циклів: а – цикл $\mathbf{1, 2-3-4-\overline{1}-\overline{2}-\overline{3}-\overline{4}}$; б – цикл $\mathbf{1-2-3-\overline{2}-\overline{3}-2-\overline{1}-4-\overline{2}-\overline{4}}$;

Складаємо умови переходу між циклами:

Нехай як і раніше у нас буде **STEP 0**, в якому система переводиться в початковий стан, однак для виходу на **STEP 1** необхідна додаткова умова - натиснута кнопка **S3**. Умова переходу в **STEP 1** зі **STEP 0** має наступний вигляд:

$$\mathbf{STEP\ 1} = \boldsymbol{XN1 \cdot XN2 \cdot XN3 \cdot XN4 \cdot S3}\,. \tag{9.62}$$

Умова повернення в **STEP 0** – всі приводи знаходяться в початковому положенні і відтиснута кнопка S3:

$$\mathbf{STEP\ 1} = \boldsymbol{XN1 \cdot XN2 \cdot XN3 \cdot XN4 \cdot NS3}\,. \tag{9.63}$$

Перехід в **STEP 2** зі **STEP 1** відбудеться коли всі приводи знаходяться в початковому положенні, натиснута кнопка ***S3*** та ***S2***, відтиснута кнопка ***S1***:

$$\mathbf{STEP\ 2} = \boldsymbol{XN1 \cdot XN2 \cdot XN3 \cdot XN4 \cdot S3 \cdot S2 \cdot NS1}\,. \tag{9.64}$$

Перехід в крок **STEP 1** зі **STEP 2** відбудеться за іншою умовою (**STEP 1***), коли всі приводи знаходяться в початковому положенні, натиснута кнопка ***S3*** та ***S1***, а також відтиснута кнопка ***S2***:

$$\mathbf{STEP\ 1*} = \boldsymbol{XN1 \cdot XN2 \cdot XN3 \cdot XN4 \cdot S3 \cdot S1 \cdot NS2}\,. \tag{9.65}$$

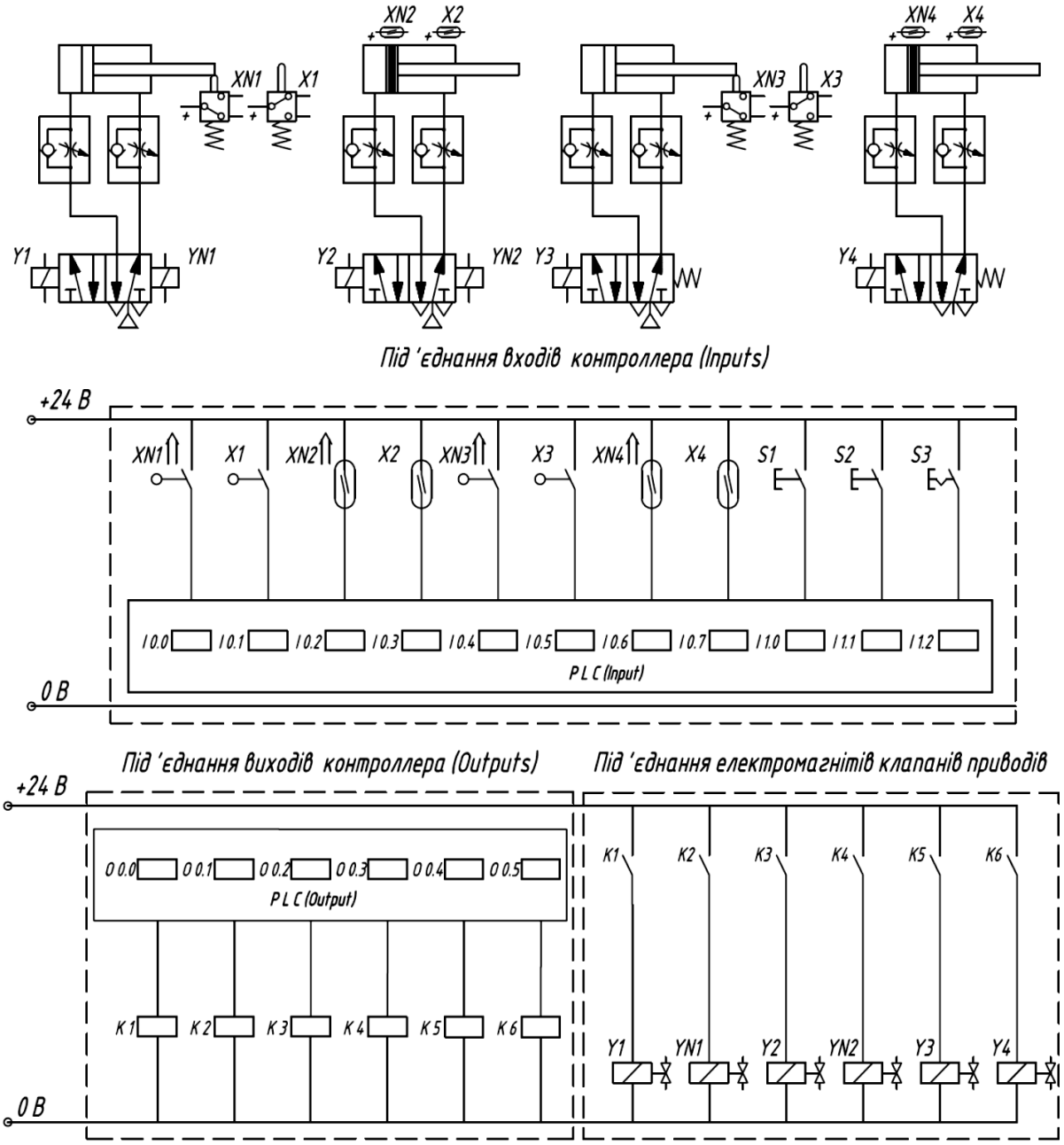


Рис. 9.47. Комбінована схема з контролером для виконання циклової мехатронної системи з двома альтернативними циклами: циклом $\mathbf{1, 2-3-4-\overline{1}-\overline{2}-\overline{3}-\overline{4}}$ та циклом $\mathbf{1-2-3-\overline{2}-\overline{3}-2-\overline{1}-4-\overline{2}-\overline{4}}$

потрібно скласти з урахуванням таблиці 9.1.

Створення програми в FST4.21 по складеним логічним командам з урахуванням Allocation List (табл. 9.1):

```
'Підготовка системи до пуску. Приведення приводів у
'початкове положення та вимикання усіх елементів
'пам'яті, які використовуються в програмі

STEP 0
 IF                    NOP
 THEN  SET             YN1
       RESET           Y1
       SET             YN2
       RESET           Y2
       RESET           Y3
       RESET           Y4
       RESET           EP1
       RESET           EP2

'Для переходу в робочий крок № 1 необхідно щоб всі
'поршні циліндрів були втягнутими і натиснена кнопка
'«Пуск S3»

 IF                    XN1             ' умова (9.62)
       AND             XN2
       AND             XN3
       AND             XN4
       AND             S3
THEN  JMP TO 1

'Далі записуємо програму згідно складених команд, не
'забуваючи про умови переходу в інші кроки

STEP 1
 IF                    S3              ' умова (9.64)
       AND     N       S1
       AND             S2
       AND             XN1
       AND             XN2
       AND             XN3
       AND             XN4
 THEN  JMP TO 2
```

```
 IF              N       S3              ' умова (9.63)
       AND               XN1
       AND               XN2
       AND               XN3
       AND               XN4
 THEN  JMP TO 0
 IF                      S1
       AND               XN4
 THEN  SET               Y1
       RESET             YN1
       SET               Y2
       RESET             YN2
 IF                      X1
       AND               X2
 THEN  SET               Y3
 IF                      X3
 THEN  SET               Y4
 IF                      X4
 THEN  SET               YN1
       RESET             Y1
 IF                      XN1
       AND               X4
       AND               X3
 THEN  SET               YN2
 RESET                   Y2
 IF                      XN2
 THEN  RESET             Y3
 IF                      XN3
 THEN  RESET             Y4
 IF                      NOP
 THEN  JMP TO 1

STEP 2
 IF                      S3              ' умова (9.65)
       AND               S1
       AND     N         S2
       AND               XN1
       AND               XN2
```

```
       AND             XN3
       AND             XN4
 THEN  JMP TO 1
 IF              N     S3
       AND             XN1
       AND             XN2
       AND             XN3
       AND             XN4
THEN  JMP TO 0
 IF                    S2
       AND             XN4
       AND             XN1
 THEN  SET             Y1
       RESET           YN1
 IF                    X1
        OR             XN3
 THEN  SET             Y2
       RESET           YN2
 IF                    X2
       AND             XN4
       AND             X1
 THEN  SET             Y3
 IF                    X3
       AND             X1
       AND             XN4
 THEN  SET             YN2
       RESET           Y2
 IF                    XN2
       AND             X1
       AND             X3
 THEN  SET             Y4
 IF                    X4
       AND             XN2
 THEN  SET             YN1
       RESET           Y1
 IF                    XN1
       AND             X3
       AND             X4
```

```
 THEN   SET             Y2
        RESET           YN2
 IF                     X2
        AND             XN1
 THEN   RESET           Y3
 IF                     XN3
        AND             XN1
        AND             X4
 THEN   SET             YN2
        RESET           Y2
 IF                     XN2
        AND             XN3
THEN    RESET           Y4

'Останній крок програми необхідно завершувати
'командою на переведення системи в інший або в цей же
'крок

IF      NOP
THEN    JMP TO 2
```

Завдання для самостійної роботи. Створити додаткову технологічну операцію в існуючій програмі для циклу $\mathbf{1.2.3.4-\overline{1}-\overline{2}-\overline{3}-\overline{4}}$, яка виконується якщо в нульовому кроці одночасно натиснути кнопки S1 та S2 і не натиснута кнопка S3. Перехід з додаткового кроку в робочий – після відпрацювання циклу додаткового кроку та натисненні кнопки S3.

9.13. **Типові операнди, оператори і команди**

Операнд має двоєдине призначення і застосування. По-перше, ***операнд*** – ідентифікатор оператора (функції) процесу виконання команди програми програмованим логічним контролером (*PLC*). По-друге, ***операнд*** є тимчасовим фізичним електронним елементом, а саме входом/виходом контролера, одно бітовим та/або мульті бітовим регістром контролера, таймером або лічильником контролера. Операнди запрошується (адресується) та виконується командами програми.

По кількості розрядів регістрів, в яких тимчасово зберігається двоїна інформація (1/0) на фізичному рівні контролера ОПЕРАНДИ можуть бути одно бітовими (ОБ-операнди), які займають один розряд (один біт) в регістрі або в акумуляторі і мульті бітовими (МБ-операнди), які займають 16 розрядів або СЛОВО (позначається першою літерою **W** англомовного **W**ord - СЛОВО) або 8 розрядів (байт) у відповідних регістрах. 32-розрядні і 64-розрядні контролери працюють з подвійними словами.

Типи і кількість операндів залежить від моделі контролера і фірми виробника контролерів. У навчальному процесі КНУТД та інших закладах вищої освіти можуть застосовуваися контролери фірми Festo (Австрія). Технічна характеристика контролерів Festo (Німеччина) і типи операндів для мови програмування **STL** наведені в таблиці 9.2.

Таблиці 9.2 Типи контролерів Festo

Тип операндів мови STL	Тип контролерів Festo			
	FPC101	**FPC202C**	**FPC404**	**FPC405**
Таймери (**T**)	32	32	16	64
Лічильники (**C**)	16	32	16	64
Флаги (**F**)	256	256	256	1024
Слово (**W**)	16x16	16x16	16x16	64x16
Регістри (**R**)	64	64	64	64
Багатозадачність	-	+	+	+
Кількість задач	1	2	4	64

Однобітові операнди (ОБ-операнди) зведені в таблицю 9.3. ВИРАЗИ (шаблони) програм складаються з двох частин - умовної і виконавчої. В **умовній частині** ВИРАЗУ, яка починається з команди «**IF**» (якщо) операнди перевіряються на «true»/«false» (1/0) – поширена назва «істина »/«хибність ». У **виконавчій частині** ВИРАЗУ, яка починається з команди «**THEN**» (тоді) над ОБ-операндами командами «**SET»/«RESET»** виконується відповідна дія – «включити»/«виключити», «встановити»/«скинути» що відповідає діям «включення/виключення» виконавчого механізму (електромагніту, електродвигуна тощо). На час операцій запиту і завантаження ОБ-операнди зберігаються в одно бітовом акумуляторі (ОБ-акумуляторі). Ідентифікатор операндів позначається в програмі першою літерою назви операнда в англомовному напису.

Таблиця 9.3 Однобітові операнди

ОБ-операнд	Іден-тифі-катор	Син-так-сис	Частина ВИРАЗУ програми	Типовий приклад фрагменту ВИРАЗУ програми	
Вхід	**I**	**I**n.n	умовна	**IF I1.0**	Якщо Вхід **1.0** активний
Вихід	**O**	**O**n.n	умовна	**IF O.1.1**	Якщо Вихід **1.1** включений
Вихід	**O**	**O**n.n	виконавча	**SET O.1.5**	Включити Вихід **1.5**
Таймер	**T**	**T**n	умовна	**IF T3**	Якщо запущений
Таймер	**T**	**T**n	виконавча	**SET T5**	таймер **3** Включити таймер **5**
Лічильник	**C**	**C**n	умовна	**IF C1**	Якщо запущений
Лічильник	**C**	**C**n	виконавча	**SET C4**	лічильник **1** Включити лічильник **4**
Флаг	**F**	**F**n.n	умовна	**IF F.1.1**	Якщо флаг **1.1** включений
Флаг	**F**	**F**n.n	виконавча	**RESET F.1.4**	Виключити флаг **1.4**

Окрім однобітових операндів наведених в табл. 9.3 в деяких моделях контролерів можуть також застосовуватися наступні ОБ-операнди із збереженням змісту в перших трьох стовпчиках таблиці 9.4:

Програма (**P**rogram) → **P** → **P**n;

Процесор (Procesor) → **Y** → **Y**n;

Стан похибки (**E**rror) → **E** → **E**;

Перезапуск (Restarting) → **ARU** → **ARU**.

Мультібітові операнди (МБ-операнди) – табл. 9.4 в умовна частині ВИРАЗУ «**IF**» перевіряються на величину (більше, менше, рівно), діапазон (0…255, 0…65535, +/- 32767) або порівнюються з іншими МБ-операндами. У виконавчій частині ВИРАЗУ мультібітові операнди могутть загружатися

«**LOAD**» значенням, збільшуватися на одиницю «**INC**», зменшуватися на одиницю «**DEC**» та використовувати інші арифметичні та логічні оператори. На час операцій запиту і завантаження МБ-операнди зберігаються у мультібітовому акумуляторі (МБ-акумуляторі).

Таблиця 9.4 Мультібітові операнди

МБ-операнд	*Ідентифікатор*	*Синтаксис*	*Частина ВИРАЗУ програми*	*Типовий приклад фрагменту ВИРАЗУ програми*	
СЛОВО ВХОДУ (**I**nput **W**ord)	**IW**	**I**Wn	умовна	IF (IW2=V237)	якщо вхідне слово **2** дорівнюється числу **237**
СЛОВО ВИХОДУ (**O**utput **W**ord)	**OW** **OW**	**OW**n **OW**n	умовна виконавча	IF (OW1=V7) LOAD V127 TO OW2	якщо слово виходу **1** дорівнюється числу **7** завантажити число **127** і перейти до слова виходу **2**
СЛОВО ТАЙМЕРА (**T**imer **W**ord)	**TW** **TW**	**TW**n **TW**n	умовна виконавча	IF (TW2<V300) LOAD V200 TO TW3	якщо слово таймера **2** < 3 секунд Завантажити число **200** до слова таймера **3**
ПРЕселектор ТАЙМЕРА (**T**imer **P**reselector)	**TP** **TP**	**TP**n **TP**n	умовна виконавча	IF (TP1<V300) LOAD V500 TO TP2	Якщо час затримки таймера **1**<3 с Завантажити число **500** до преселектора таймера **2**

СЛОВО ЛІЧИЛЬника (**C**ounter **W**ord)	**CW** **CW**	**CW**n **CW**n	умовна виконавча	IF (CW3<>V30) THEN INC CW2	Якщо Слово Лічильника **1** не рівно числу **30** Тоді збільшити на **1** вміст слова лічильника **2**
ПРЕселектор ЛІЧИЛЬника (**C**ounter **P**reselector)	**CP** **CP**	**CP**n **CP**n	умовна виконавча	IF (CP1=V150) LOAD V15 TO CP3	якщо преселектор лічильника **1** рівний числу **150** завантажити число **15** до ПРЕселектора лічильника **3**
РЕГІСТР (**R**egister)	**R** **R**	**R**n **R**n	умовна виконавча	IF(R20 = V23)I LOAD (R25) TO R27	Якщо флаг **1.1** включений Завантажити вміст регістра **25** до регістра **27**
Слово помилки **(E**rror **W**ord**)**	**EW** **EW**	**EW**n **EW**n	умовна виконавча	IF(EW V15) LOAD V0 TO EW	Якщо слово помили **EW** э число 15 Завантажити число «0» в **EW**

О п е р а т о р и. Для побудови ВИРАЗІВ, в мові STL використовують наступні оператори і умовні символи, які наведені в таблиці 9.5.

Таблиця 9.5 Умовні символи

N	НІ (заперечення)
V,V$,V%	Завдання ЗНАЧЕННЯ для МБ-операнди в десятинному, 16-річному і двійковому форматі, відповідно
+, - , *, /	Складання, віднімання, множення та ділення МБ-операндів і констант
< , > , = , <> , <= , >=	Мультібітове порівняння: менше ніж, більше ніж, рівно, дорівнює, не дорівнює, менше або дорівнює, більше або дорівнює

Типові команди ВИРАЗІВ програм

В основу програмування на мові STL цикловими системами керування цільовими механізмами і пристроями технологічних машин галузі покладена структура ВИРАЗІВ, які складаються з двох частин «*умовній*» і «*виконавчій*», які починаються з команди «**Якщо** …» і команди «**Тоді** … » або в англомовному напису є командою «**IF** …» і командою «**THEN** … ». Надалі наведені типові команди ВИРАЗІВ програм і приклади їх застосування у фрагментах програм.

STEPn (крок) **–** ця команда позначає початок ВИРАЗУ

IF (*якщо*) **–** ця команда позначає початок умовної частини ВИРАЗУ;

THEN (тоді, потім) **–** команда позначає початок виконавчій частини ВИРАЗУ.

SET – (*від* **SET**ting *включити, перевести в активний стан, встановити*) **–** команда використовується для переведення ОБ-операндів в стан логічної одиниці («1»-стан). Дія цієї команди залежить від адресуємого операнда згідно з приклади зведеним в таблицю 9.6.

Таблиця 9.6 Приклади застосування команди **SET**

ОБ-операнди	*Синтаксис*	*Пояснення дії*
ВИХІД	**SET O01.1**	Включити ВИХІД 1.1
ФЛАГ	**SET F 01.3**	Встановити ФЛАГ в стан «1»
ТАЙМЕР	**SET T1**	Активний стану ТАЙМЕРУ 1 або ТАЙМЕР 1 включений в роботу.

		При цьому вміст преселектора ТР1 (початкової установки часу затримки) для таймера 1 автоматично адресується (копіюється) в слово таймера **1** (**TW1**)
ЛІЧИЛЬНИК	**SET C1**	Активний стан ЛІЧИЛЬНИКА 1 або ЛІЧИЛЬНИК 1 включений в роботу. При цьому вміст преселектора С1 (початкової установки часу затримки) для ЛІЧИЛЬНИКА 1 автоматично адресується (копіюється) в СЛОВО ЛІЧИЛЬНИКА 1 (**CW1**)

RESET (*виключити, перевести в пасивний стан*) **–** команда використовується для переведення одно бітових операндів в стан логічного нуля («0» - стан).

Приклади застосування команди **RESET***:*

Операнди	*Синтаксис*	*Пояснення дії*
ВИХІД	**RESET O01.1**	Виключити ВИХІД 01.1
ФЛАГ	**RESET F01.3**	Перевести статус ФЛАГу 01.3 у «0»-стан
ТАЙМЕР	**RESET T1**	Стан ТАЙМЕРА Т1=0 (вимкнений)

OTHRW (*від* **OTH**e**RW**ise - інакше) – команда забезпечує продовження виконання програми, Якщо умовна частина ВИРАЗУ хібна.

Приклад застосування команди **OTHRW***:*

Операнди	*Синтаксис*	*Пояснення дії*
ВХІД	**IF I1.0**	ЯКЩО ВХІД 1.0 активний
ВИХІД	**THEN SET O2.1**	ТОДІ ВКЛЮЧИТИ ВИХІД 2.1
ВИХІД	**OTHRW SET O2.2**	ІНАКШЕ ВКЛЮЧИТИ ВИХІД 2.2

AND (і) – команда операції логічного складання (операція логічне «И» або логічна операція диз'юнкція). Команда **AND** має поширене застосування у двох призначеннях:

а – з використанням операції логічного множення «И»;

б – для виконання операції логічного множення «И» над двома МБ-операнди або значеннями, як в умовної, так і у виконавчій частинах ВИРАЗУ.

Приклад застосування команди **AND** для поєднання двох або більше ОБ-операнди або МБ-операнди в умовної частині ВИРАЗУ

Операнд	*Синтаксис*	*Пояснення дії*
ВХІД	**IF I1.1**	Якщо ВХІД 1.1 активний
ЛІЧИЛЬНИК	**AND C1**	і ЛІЧИЛЬНИК С1 включений
ВИХІД	**THEN SET O2.5**	ТОДІ ВКЛЮЧИТИ ВИХІД 2.5

Приклад застосування команди **AND** за умовою виконати побітове операцію множення двох 8-бітових операндів з третім операндом результату для три адресної команди при використанням трьох операндів і регістрів R7, R8 та R9:

10101010 **операнд 1** (8-бітовий регістр R7) = 170 десятинне

00001111 **операнд 2** (8-бітовий регістр R8) = 15 десятинне

00001010 **результат** (8-бітовий регістр R9) = 10 десятинне

Операнд	*Синтаксис*	*Пояснення дії*
РЕГІСТР	**IF (R7 =V170)**	Якщо істинне, що вміст регістру 7 рівний числу 170
РЕГІСТР	**AND (R8=V15)**	і вміст регістру 8 рівний числу 15
РЕГІСТРИ	**THEN LOAD (R7 AND R8) TO R9**	ТОДІ ЗАВАНТАЖИТИ вміст регістру 7 в МБ акумулятор, виконати операцію логічного множення «И» із змістом двох регістрів 7 та 8 і результат помістити в регістр 9

OR (або) – команда операції логічного множення (операція логічне «ИЛИ» або операція кон'юнкція) виконує логічне «ИЛИ» над одно бітовими і мульті бітовими операндами.

NOP – спеціальна команда, яка має застосування у двох призначеннях:

Приклад застосування команди **NOP** ***в умовній частині*** ВИРАЗУ - має значення «*завжди істинність*» і викликає безумовне виконання той дії, яка передбачена у виконавчій частині що починається з команди THEN. Така ініціалізація потрібна при подачі живлення в систему керування на першому кроці програми:

```
IF      NOP              /ВИКОНУВАТИ ЗАВЖДИ

THEN    LOAD    V0       /ЗАВАНТАЖИТИ ЧИСЛО 0

TO      OW1              /В СЛОВО ВИХОДУ 1, тобто в
                         / цьому випадку команда NOP
                         /забезпечує ВИМИКАННЯ ВИХОДУ 1
```

Приклад застосування команди **NOP** ***у виконавчій частині*** ВИРАЗУ - має значення «*нічого не робити*» у випадку очікування в програмі певних умов, а також для переходу до наступного кроку програми.

```
IF      I.1.1            /ЯКЩО ВХІД 1.1 активний

THEN    NOP              / ТОДІ НІЧОГО НЕ РОБИТИ
                         / і перейти до наступного кроку програми
```

JMP – скорочена назва команди *від анл.* **J**u**MP** (стрибок**).** Ця команда служить для переривання програмного потоку при організації циклу (циклів) всередині програми, а також встановлення пріоритету між ВИРАЗАМИ.

Приклади застосування команди **JMP:**

STEP 10		
Вирази	*Синтаксис*	*Пояснення дії*
ВИРАЗ 1	**IF I.1.0** **THEN SET O2.1** **JMP TO 15**	Якщо ВХІД **1.0** Активний тоді включити ВИХІД **2.1** і перейти до кроку **15**
ВИРАЗ 2	**IF N I.1.3** **AND I.1.5** **THEN RESET O.2.2** **JMP TO 7**	Якщо ВХІД 1.3 неактивний І ВХІД **1.5** АКТИВНИЙ ТОДІ ВИМКНУТИ ВИХІД **2.2** і ПЕРЕЙТИ до кроку **7**
ВИРАЗ 3	**IF T1** **THEN SET F0.0**	Якщо ТАЙМЕР **1** включений ТОДІ ВСТАНОВИТИ флаг **F0.0**
ВИРАЗ 4	**IF NOP** **THEN SET O.2.3**	ЗАВЖДИ ВКЛЮЧЕНИЙ СТАН ВИХОДУ **2.3**

З наведених прикладів для чотирьох виразів випливає що ВИРАЗИ 2, 3 і 4 обробляються тільки тоді, коли умова у ВИРАЗУ 1 хібна (=0) і нема переходу по команді **JMP** TO 15. Коли же умова у ВИРАЗУ 1 істина (=1), тобто виконується, тоді програма переходить на крок 15 і не обробляє ВИРАЗИ 2, 3 і 4 в кроці 10.

Інші деякі поширені команди:

DEC (від **DEC**rement **–** зменшення) – команда зменшує на одиницю величину любого МБ-операнда. Операція зменшення операнда на 1 (арифметична операція віднімання одиниці) відбувається безпосередньо над МБ-операндом без попереднього його завантаженні в МБ-акумулятор, що важливо для застосування цієї команди при програмування ЛІЧИЛЬНИКІВ.

INC (від **INC**rement **–** збільшення) – команда збільшує на одиницю величину любого МБ-операнда. Операція збільшення операнда на 1 (арифметична операція складання з одиницею) відбувається безпосередньо над МБ-операндом без попереднього його завантаженні в МБ-акумулятор, що важливо для застосування цієї команди при програмуванні ЛІЧИЛЬНИКІВ.

SHL – мультібітова команда (МБ-команда), яка зсовує все біти МБ-акумулятора на один розряд (позицію) вліво, при цьому найбільше значущий біт (крайній лівий розряд) втрачається, а в найменше значущий біт (крайній правий розряд) записується "0". Напис команди складається від скорочення двох англомовних слів – перші дві літери **SH** команди **SH**L утворені від слова «**CH**ange» (зсув), а літері **L** команди SH**L –** відповідає Слово «the **L**eft» (вліво). Тому команда **SHL** призначена для зсуву вліво на одну позицію вмісту багатобітового акумулятора. Команда **SHL** застосовується для утворення зсувного регістру, а також при множенні багатобітового операнда або числа на два. Використання команди двічі підряд **SHL SHL** виконує множення на чотири і так далі.

SHR– МБ-команда, яка зсовує все біти МБ-акумулятора на один розряд вправо, при цьому найменше значущий біт (крайній правий розряд) втрачається, а в найбільше значущий біт (крайній лівий розряд) записується "0".

ROL – МБ-команда, яка зсовує все біти МБ-акумулятора на *один розряд* вліво, при цьому найбільше значущий біт (крайній лівий розряд) пересувається в найменше значущий біт (крайній правий розряд).

ROR– МБ-команда, яка зсовує вправо все біти МБ-акумулятора на один розряд вправо, при цьому найменше значущий біт (крайній правий розряд) пересувається в найбільше значущий біт (крайній лівий розряд).

ЗМІСТ

Список літератури

1. Губарев А.П. Мехатроника: от структуры системы к алгоритму управления: учебное пособие / А.П. Губарев, О.В. Левченко. – К.: НТУУ «КПИ». – 2007. – 180 с.

2. Губарев А.П. Дискретно-логическое управление в системах гидропневмоавтоматики: учебное пособие. – К.: ИСМО. – 1997. – 224 с.

3. Губарев О.П. Мехатроніка: циклічно-модульний підхід до вирішення практичних задач автоматизації / О.П. Губарев, О.С. Ганпанцурова. – К.: НТТУ «КПІ». – 2016. – 160 с.

4.Camozzi. Пневматика для всех. От теоретических основ к практическим навыкам. – C_Competence. –ТОВ «Камоцці».– didactic@camozzi.ua

5. Орловський Б.В. Електронний конспект лекцій з дисципліни «Мехатроніка в галузевому машинобудуванні». – К.: КНУТД, msnp. – Конспект лекцій.

6. Орловський Б.В. «Мехатроніка в галузевому машинобудуванні». - Методичні вказівки та завдання до лабораторних робіт для студентів денної та заочної форми спеціальності «Галузеве машинобудування» галузі знань «Механічна інженерія», ступеня вищої освіти - «Бакалавр» / Б.В. Орловський, Ю.О. Цибрій. – К.: КНУТД. – 2016. – 52 с.

7. Яхно О.М. Введение в мехатронику / О.М. Яхно, А.В. Узунов, А.Ф. Луговской и др. – К.: НТУУ «КПИ». – 2008. – 528 с.

8. Пашков Е.В. Промышленные мехатронные системы на основе пневмопривода: учебное пособие / Е.В. Пашков., Ю.А. Осинский. – Севастополь: Изд-во СевНТУ, 2007. – 401 с.

9. Орловський Б.В. Мехатроніка в галузевому машинобудуванні. - Методичні вказівки та завдання для самостійної роботи студентів денної і заочної форми навчання галузі знань 0505 «Машинобудування та матеріалообробка», напряму підготовки 6.050502 «Інженерна механіка», 6.050503 «Машинобудування» - освітньо-кваліфікаційного рівня «Бакалавр» / Б.В. Орловський. – К.: КНУТД, 2015. – 32 с.

10. Орловський Б.В. CALS-технології об’єктно-орієнтованого проектування і виготовлення взуття на засадах програмного комплексу DelCAM Crispin / Б.В. Орловський. – К.: Вісник КНУТД, №1, 2012, с. 22-33.

11. ГОСТ 2.701-84. Схемы. Виды и типы. Общие требования к выполнению.

12. ГОСТ 2.781-96 ЕСКД. Обозначения условные графические. Аппараты гидравлические и пневматические, устройства управления и приборы контрольно-измерительные.

13. ГОСТ 2.702-75 ЕСКД. Правила выполнения электрических схем.

14. ГОСТ 2.722-68 ЕСКД. Обозначение условные графические в схемах. Машины электрические.

15. Пищиков В.О. Синтез багатокрокових кулачкових програмоносіїв швейних машин-напівавтоматів / В.О.Пищиков, Б.В. Орловський. – Вісник КНУТД, №5 (т. 2), 2010, с. 107-114.

16. Пищиков В.О. Проектування швейних машин: навчальний посібник для ВНЗ / В.О. Пищиков, Б.В. Орловський. – К.: Видавничо-поліграфічний дім «Формат». – 2007. – 329 с.

17.Орловський Б.В. Плосков'язальні машини (комп'ютерні, напівавтоматизовані, ручні). Конструкція та сервісне обслуговування: навчальний посібник / Б.В. Орловський, В.М. Дворжак. – К.: КНУТД. – 2012. – 247 с.

18.Орловський Б.В. Інтегрування систем циклового мехатронного керування у середовище систем керування типу «розподільний вал» технологічних машин легкої промисловості / Б.В. Орловський, В.М. Дворжак. - Вісник КНУТД, №3, Спеціальний випуск. – 2013. – с. 242-247.

19.Орловський Б.В. Технологічне обладнання галузі: навчальний посібник / Б.В. Орловський, Н.С. Абрінова. – К.: КНУТД. – 2013. – 285 с.

20.Орловский Б.В. Роботизация швейного производства / Б.В. Орловский. – К.: Техніка. – 1986. – 160 с.

21.Волков Ю.Д. Программируемые контроллеры «Фесто». – К.: Изд-во ДП «Фесто». – 2003. – 94 с.

22. Бадд Тимоти. Об'єктно-ориентированное программирование в действії / перев. с англ.: - СПб.: Питер. – 1997. – 464 с.

23. Буч Гради. Об'єктно-ориентированний анализ и проектування с примерами приложений на С++, 2ое изд./Пер..с англ.:-СПб.: "Невский диалог". – 1999. – 560 с.

24. Гамма Эрих. Приёми объектно-ориентированного проектирования. Патерны проектирования / Э. Гамма, Э. Хелм, Р. Джонсон, Д. Влиссидес. – СПб.: Питер.– 2015. – 368 с.

25. Дзюба В.И. Научные основы автоматизированного проектирования рабочих процессов трикотажных машин (объектно-ориентированный подход): монография. – К.: КГУТД. – 2000. – 186 с.

26. Орловський Б.В., Тропша Д.А. Основные принципы объектно-ориентированного проектирования рабочих процессов и машин лёгкой промышленности. – К.: Вісник ДАЛПУ, №2, 2000. – с. 44-51.

27. Орловський Б.В., Поповиченко С.А. Об'єктно-орієнтований аналіз і синтез циклового програмного керування автоматизованим завантажувальним пристроєм взуттєвих машин. – Вісник Хмельницького національного університету, №3, 2013, с.182-189.

28.Орловський Б.В. Основы автоматизации швейного производства // учебник (2ое издание) / Б.В. Орловський.– М.: Легпромбытиздат. – 1988. – 248 с.

29.Орловський Б.В. Аналіз і розрахунок механізмів переміщення одноголових напівавтоматів з числовим програмним керуванням / Б.В. Орловський, Г.В. Кошель, В.Б. Мачульський. – Вісник КНУТД, 2010, №5 (т. 2), с. 91-96.

30.Орловський Б. В. Вимірювальні засоби до виконання випускних магістерських робіт: Методичні вказівки для самостійної роботи магістрантів спеціальності 8.05050316 «Обладнання легкої промисловості та побутового обслуговування» / Б.В. Орловський, Г.В. Кошель, В.М. Дворжак.– К.: КНУТД. – 2011. – 59 с.

31. Галінковський К.В. Практична електроніка: навчальний посібник / К.В.Галінковський, Л.В.Грузд, А.Ф.Ластовець, С.О.Ластовець. - К.: Серія конструктор №7, 2017. – 90 с.

32. Губарев О.П. Структурно-модульний синтез циклових систем гідро- та пневмоприводу: автореф. дис... д-ра техн. наук: 05.02.03 / О.П. Губарев. - Національний технічний ун-т України "Київський політехнічний ін-т". – К.: 2004. – 35 с.

33. Якимчук М.В. Науково-технічні засади створення обладнання для групового пакування харчових продуктів на основе мехатронних модулів: автореф. дис... д-ра техн. наук: 05.18.12 / М.В. Якимчук. - Національний університет харчових технологій. – К.: 2016. – 38 с.

Навчальне видання

Орловський Броніслав Вікентійович

МЕХАТРОНІКА В ГАЛУЗЕВОМУ МАШИНОБУДУВАННІ

Навчальний посібник

Редактор *Л. Л. Овєчкіна*
Відповідальний за поліграфічне видання *Ю. В. Коноваленко*
Коректор *Н. П. Біланюк*

Підп. до друку 29.11.2017 р. Формат 60х84 1/16.
Ум. друк. арк. 24,17. Облік. вид. арк. 18,93. Тираж 20 пр. Зам. 335.

Видавець і виготовлювач Київський національний університет технологій та дизайну.
вул. Немировича-Данченка, 2, м. Київ-11, 01011.

Свідоцтво суб'єкта видавничої справи ДК № 993 від 24.07.2002.