Хмельницький національний університет

МЕТОДИЧНІ ВКАЗІВКИ З ДИСЦИПЛІНИ «АВТОМАТИЗАЦІЯ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ» ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ: «РОЗРОБКА ПРОГРАМИ КЕРУВАННЯ ТЕХНОЛОГІЧНИМ ПРОЦЕСОМ АВТОМАТИЗОВАНОГО ПЕРЕМІЩЕННЯ ПРОДУКЦІЇ»

для здобувачів вищої освіти освітніх рівнів: бакалавр зі спеціальності 208 «Агроінженерія» та магістр зі спеціальностей 133 «Галузеве машинобудування» та 141 «Електроенергетика, електротехніка та електромеханіка»

Затверджено на засіданні кафедри машин і апаратів, електромеханічних та енергетичних систем. Протокол № 6 від 03.12.2020

Хмельницький 2020

Методичні вказівки з дисципліни «Автоматизація технологічних процесів» для здобувачів вищої освіти освітніх рівнів бакалавр спеціальності 208 «Агроінженерія» та магістр спеціальностей 133 «Галузеве машинобудування», 141 «Електроенергетика, електротехніка та електромеханіка» для виконання лабораторних робіт: Розробка програми керування технологічним процесом автоматизованого переміщення продукції / С.Л. Горященко, М.В. Лук'янюк, П.С. Майдан, А.О. Поліщук – Хмельницький : ХНУ, 2020. – 18 с.

Укладачі
Горященко С.Л., канд. техн. наук, доц.;
Лук'янюк М.В., канд. техн. наук, доц.;
Майдан П.С., канд. техн. наук, доц.;
Поліщук А.О., асист.

Відповідальний за випуск: Поліщук О.С., докт. техн. наук, доц.

Редактор-коректор: Яремчук В. С.

Технічне редагування, коректування і верстка: Чопенко О. В.

Макетування та друк здійснено редакційно-видавничим центром Хмельницького національного університету (м. Хмельницький, вул. Інститутська, 7/1). Підп. до друку _____. Зам. № _____, електронне видання, 2020.

ХНУ, 2020

ВСТУП

Дисципліна «Автоматизація технологічних процесів» є однією із фахових дисциплін і займає провідне місце у підготовці здобувачів вищої освіти освітніх рівнів бакалавр спеціальності 208 «Агроінженерія» та магістр спеціальностей 133 «Галузеве машинобудування», 141 «Електроенергетика, електротехніка та електромеханіка».

Методичні вказівки для виконання лабораторної роботи: «Розробка програми керування технологічним процесом автоматизованого переміщення продукції».

Мета дисципліни. Навчити здобувачів вищої користуватися сучасним програмним забезпеченням, із використанням сучасних систем автоматизації технологічних процесів фірми Siemens.

Предмет дисципліни. Програмне забезпечення, з використанням сучасних систем автоматизації технологічних процесів фірми Siemens.

Завдання дисципліни. Формування практичних навичок із використання програмного забезпеченням фірми Siemens для автоматизації технологічних процесів.

Відповідно до Стандартів вищої освіти із зазначених спеціальностей та освітніх програм дисципліна має забезпечити:

Результати навчання. Здобувач вищої освіти освітнього рівня бакалавр, який успішно завершив вивчення дисципліни, повинен:

- володіти природничо-науковими та професійними знаннями; виявляти, узагальнювати та вирішувати проблеми, що виникають у процесі професійної діяльності; виконувати експериментальні дослідження роботи сільськогосподарської техніки в конкретних умовах використання; вибирати машини і обладнання та режими їх роботи у механізованих технологічних процесах рослинництва, тваринництва, первинної обробки сільськогосподарської продукції; проектувати технологічні процеси та обґрунтовувати комплекси машин для механізованого виробництва сільськогосподарської продукції; розробляти операційні карти для виконання механізованих технологічних процесів; застосовувати закони електротехніки для пояснення будови і принципу дії електричних машин; визначати параметри електроприводу машин і обладнання сільськогосподарського призначення; вибирати і використовувати

системи автоматизації та контролю технологічних процесів в аграрному виробництві (208).

Здобувач вищої освіти освітнього рівня магістр, який успішно завершив вивчення дисципліни, повинен:

- вміти проводити аналіз стану експлуатації, приймати рішення щодо підвищення надійності роботи, планувати модернізації, ремонт та заміну обладнання; здійснювати складні вимірювання електричних параметрів устаткування, обробку та аналіз їх результатів із застосуванням сучасної контрольно-вимірювальної та обчислювальної техніки; збирати, обробляти та накопичувати вихідні матеріали, дані статистичної звітності (141).

- приймати участь в технологічній підготовці виробництва виробів різного призначення та принципу дії; розробляти технічне завдання на проектування та виготовлення машин, приводів, систем та нестандартного обладнання та засобів технологічного оснащення; враховувати сучасні тенденції розвитку техніки та технологій; розраховувати та проектувати елементи та пристрої, засновані на різноманітних фізичних принципах дії; організовувати роботи по вдосконаленню, модернізації, уніфікації обладнання, проводити заходи по створенню системи якості (133).

1 ЗАГАЛЬНІ ВИМОГИ ТА РЕКОМЕНДАЦІЇ

1.1. Середовище розробки **TIA Portal V15.1.**

Головними аспектами нової версії є можливість конфігурації нових резервованих контролерів **SIMATIC S7-1500R** і **SIMATIC S7-1500H**, ізохронний режим на центральній шині **S7-1500**, керування доменами **MRP** за межами проекту та інші.

Інженерні опції:

TIA Portal Multiuser Engineering призначено для одночасної роботи декількох користувачів над одним проектом. Це означає, що можна значно зменшити час конфігурації, в результаті чого проекти швидше вводяться в експлуатацію.

TIA Portal Teamcenter Gateway призначений для збереження та керування проектами **TIA Portal**, а також глобальними бібліотеками в **Teamcenter**. Керування оператором інтегровано в **TIA Portal**.

TIA Portal Cloud Connector призначений для здійснення доступу до **TIA Portal** локальних комп'ютерів і підключеного обладнання **SIMATIC**. Робота в **TIA Portal** може вестися з віддаленого робочого столу в приватній хмарі.

TIA User Management Component надає можливість глобального адміністрування користувачів. Користувачі і групи користувачів можуть бути визначені і керуватись між проектами. Користувачі і групи користувачів можуть бути взяті з **Microsoft Active Directory**.

TIA Portal Openness. Користувачі можуть використовувати **API**-інтерфейс в **WinCC** і **STEP 7** в **TIA Portal** для інтеграції **TIA Portal** в своєму середовищі розробки і автоматизації інженерно-технічних завдань. З'являється можливість писати свої власні програми із зовнішніми середовищами розробки, наприклад, генератор коду для **HMI** панелей і блоки **PLC**.

SIMATIC Energy Suite призначено для прямого об'єднання керування споживанням електроенергії та автоматики. За рахунок цього досягається прозорість споживання електроенергії на підприємстві. Крім цього, спрощене програмування вимірювального обладнання значно зменшує час на конфігурацію.

SIMATIC PLCSIM Advanced призначено для створення віртуальних контролерів щоб здійснити моделювання контролерів **SIMATIC S7-1500** і **ET 200SP** і використовувати віртуальні контролери для комплексного моделювання.

SIMATIC Target 1500S є додатком для **Simulink** від **MathWorks**. Ця програма служить для інтеграції моделей **Simulink** безпосередньо в програмний цикл **ODK** сумісних контролерів **SIMATIC S7-1500**.

SIMATIC Visualization Architect (**SiVArc**) призначений для швидкого, простого і гнучкого автоматичного створення вмісту **HMI** проектів на базі програми **STEP 7** користувача.

Runtime опції: **SIMATIC Safe Kinematics** - нова опція для безпечного контролювання руху кінематики в приміщенні. Можна безпечно контролювати швидкість деяких точок в кінематиці, таких як центральна точка інструменту (**TCP**), а також робочі та захисні зони. Наступна кінематика доступна в першій версії: маніпулятор шарнірної конструкції, **SCARA**, вертикальний підбирач крену і декартовий портал.

SIMATIC ProDiag - призначений для вибіркової та швидкої діагностики обладнання **SIMATIC S7-1500** і **SIMATIC HMI**.

SIMATIC OPC UA S7-1500 - опція, призначена для легкого підключення будь-якого стороннього пристрою до контролера **S7-1500** через **OPC UA** сервер, вбудований в контролер **S7-1500**.

SIMATIC Energy Suite S7-1500 поставляється з ліцензією для 10 енергетичних об'єктів. Додаткові енергетичні об'єкти можуть бути додані з **runtime** пакетами.

WinCC/WebUX для **WinCC Professional** призначено для контролю виробничих процесів і при необхідності для керування ними через інтернет або інтранет за допомогою мобільних пристроїв **HMI**.

1.2. Організація та порядок виконання лабораторної роботи. Роботи виконуються кожним студентом індивідуально, на своєму робочому місці, за своїм варіантом завдання. Лабораторна робота завершується оформленням звіту та її захистом і вважається зарахованою, якщо звіт містить необхідні блок-діаграми, схеми, графіки, виконані правильно та акуратно, а також, якщо студент відповів на питання викладача, показавши знання з будови та принципу роботи об’єкта досліджень і розуміння процесів, що пояснюють отримані результати. Крім того, студент повинен знати призначення всіх елементів блок-діаграм та схем і вміти пояснити порядок дій при їх складанні.

1.3. Обробка результатів дослідів та оформлення звіту. Кожен студент повинен самостійно обробити результати виконаних ним дослідів і скласти звіт з лабораторної роботи. Звіт, крім номера та назви роботи, індексу навчальної групи, повинен містити наступні

відомості:

– програма автоматизації процесу переміщення продукції та її блок-діаграма;

– опис побудови програми автоматизованого процесу переміщення продукції;

– висновки щодо роботи.

Усі блок-діаграми, таблиці, графіки, схеми, що приводяться в звіті, повинні мати найменування. При виконанні розрахунків рекомендується користуватися калькуляторами.

В останньому розділі звіту – у висновку про виконану роботу – студент повинен дати оцінку побудованому віртуальному пристрою, його властивостям. Звіт повинен бути лаконічним, але таким, щоб його зміст був зрозумілим без додаткових усних пояснень.

Обсяг звіту не повинен перевищувати чотирьох сторінок ф. А4.

1.4. Критерії оцінювання. При виконанні лабораторної роботи та захисту звіту до неї, використовують наступну систему критеріїв оцінювання.

Оцінку «***відмінно***» (шкала ECTS – A) студент отримує за глибоке і повне опанування теоретичного матеріалу; легко в ньому орієнтується і вміло використовує понятійний апарат; уміння пов'язувати теорію з практикою, вирішувати практичні завдання, впевнено висловлювати і обґрунтовувати свої судження. Відмінна оцінка передбачає грамотний, логічний виклад відповіді (як в усній, так і у письмовій формі), якісне зовнішнє оформлення роботи. Студент не вагається при видозміні запитання, вміє робити детальні та узагальнюючі висновки. При відповіді допустив дві-три несуттєві ***похибки***.

Оцінку «***добре***» (шкала ECTS – B) студент отримує за повне засвоєння теоретичного матеріалу, володіння понятійним апаратом, орієнтування у вивченому матеріалі; свідомо використовує теоретичні знання для вирішення практичних задач; виклад відповіді грамотний, але у змісті і формі відповіді можуть мати місце окремі неточності, нечіткі формулювання закономірностей тощо. Відповідь студента має будуватися на основі самостійного мислення, але з наявністю нечітких формулювань.

Оцінку «***добре***» (шкала ECTS – C) студент отримує за повне засвоєння теоретичного матеріалу, володіння понятійним апаратом, орієнтування у вивченому матеріалі; свідомо використовує теоретичні знання для вирішення практичних задач; виклад відповіді грамотний, але у змісті і формі відповіді можуть мати місце окремі неточності, нечіткі формулювання закономірностей тощо. Відповідь студента має

будуватися на основі самостійного мислення. Студент у відповіді допустив дві–три ***суттєві помилки***.

Оцінку «***задовільно***» (шкала ECTS – D) студент отримує за достатні знання основного програмного матеріалу в обсязі, необхідному для подальшого навчання та практичної діяльності за професією, справляється з виконанням практичних завдань, передбачених програмою. Як правило, відповідь студента будується на рівні репродуктивного мислення, студент має слабкі знання структури курсу, допускає неточності і ***суттєві помилки*** у відповіді, вагається при відповіді на видозмінене запитання. Разом з тим набув навичок, необхідних для виконання нескладних практичних завдань, які відповідають мінімальним критеріям оцінювання і володіє знаннями, що дозволяють йому під керівництвом викладача усунути неточності у відповіді.

Оцінку «***задовільно***» (шкала ECTS – E) студент отримує за неповне опанування теоретичного матеріалу, однак отримані знання відповідають мінімальним критеріям оцінювання; розрахунки, графіки, блок-схеми та висновки виконані з певними неточностями; звіт захищений після закінчення встановленого терміну.

Оцінку «***незадовільно***» (шкала ECTS – FX) студент отримує за розрізнені, безсистемні знання, не вміє виділяти головне і другорядне, допускається помилок у визначенні понять, перекручує їх зміст, хаотично і невпевнено викладає матеріал, не може використовувати знання при вирішенні практичних завдань.

Оцінку «***незадовільно***» (шкала ECTS – F) студент отримує за повне незнання і нерозуміння теоретичного матеріалу та невиконання роботи.

При оцінюванні використовуються різні засоби контролю, зокрема: засвоєння теоретичного матеріалу перевіряється тестовим контролем; якість виконання, набуття теоретичних знань і практичних навичок перевіряється шляхом захисту кожної лабораторної роботи.

Оцінка, яка виставляється за лабораторне заняття, складається з таких елементів: усне опитування студентів перед допуском до виконання лабораторної роботи; знання теоретичного матеріалу з теми; вільне володіння студентом спеціальною термінологією і уміння професійно обґрунтувати прийняті конструктивні рішення; своєчасний захист лабораторної роботи.

Термін захисту лабораторної роботи вважається своєчасним, якщо студент захистив її на наступному після виконання роботи занятті. Пропущене лабораторне заняття студент зобов’язаний відпрацювати в лабораторіях кафедри у встановлений викладачем термін з реєстрацією у відповідному журналі кафедри, але не пізніше, ніж за два тижні до кінця теоретичних занять у семестрі.

Лабораторна робота

Розробка програми керування технологічним процесом автоматизованого переміщення продукції

***Мета роботи*:** вивчити технологічний процес, алгоритм керування, розробити програмне та технічне забезпечення автоматизованої системи керування процесом переміщення продукції.

Порядок виконання

1. Ознайомитись з відомостями щодо елементів середовища програмування **TIA Portal V15**.
2. Створити програми, згідно завдання та запустити їх у роботу.
3. Скласти звіт і зробити висновок про виконану роботу.

Короткі теоретичні відомості

Для розробки програм для ПЛК в даний час використовуються інтегровані середовища розробки (**ICP**, англ. **IDE - Integrated development environment**), що містять в своєму складі текстові редактори, компілятори, редактори зв'язків, завантажувачі та симулятори. **ICP** зазвичай являє собою єдину програму, в якій проводиться вся розробка. Вона, як правило, містить багато функцій для створення, зміни, компілювання, розгортання і налагодження програми ПЛК.

TIA Portal (**Totally Integrated Automation Portal**) - інтегроване середовище розробки програмного забезпечення систем автоматизації технологічних процесів на основі обладнання виробництва фірми **Siemens**. У **TIA Portal** об'єднані три основних програмних пакета:

- **Simatic Step 7 V.15** для програмування контролерів **S7-1200**, **S7-300**, **S7-400** і **WinAC**;
- **Simatic WinCC V.15** для розробки людино-машинного інтерфейсу (програмування сенсорних панелей та **SCADA**-систем);
- **Sinamics StartDrive V.15** для програмування перетворювачів частоти **Sinamics**.

LAD (**Ladder Diagram**) - релейні діаграми. Редактор відображає програму в графічному поданні, схожому на електричну монтажну схему. Логічні схеми дозволяють програмі імітувати перебіг

електричного струму від джерела напруги через ряд логічних умов на входах, які активізують умови на виходах. Джерелом напруги виступає шина, яка перебуває зліва.

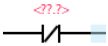
Основними елементами є нормально замкнуті і нормально розімкнені контакти.

– **Normally open contact** - Активація нормально розімкнутого контакту залежить від стану сигналу відповідного операнду. Коли операнд має стан сигналу «1», нормально розімкнутий контакт замикається, і стан сигналу на виході встановлюється на стан сигналу на вході.

Коли операнд має стан сигналу «0», нормально розімкнутий контакт не активується, і стан сигналу на виході інструкції скидається до «0».

Два або більше нормально розімкнутих контактів по послідовному з'єднанню з'єднуються покроково, реалізуючи логічну операцію «**І**». При послідовному підключенні потужність тече, коли всі контакти замикаються.

Звичайно розімкнуті контакти реалізують логічну операцію **АБО** при паралельному підключенні – тобто при паралельному підключенні сигнал проходить, коли один із контактів замкнутий.

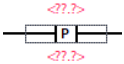


– **Normally closed contact** - Активація нормально замкнутого контакту залежить від стану сигналу відповідного операнду. Коли операнд має стан сигналу «1», нормально замкнутий контакт розмикається і стан сигналу на виході інструкції скидається до "0".

Коли операнд має стан сигналу «0», нормально замкнутий контакт не перемикається, і стан сигналу на вході передається на вихід.

Два або більше нормально замкнутих контактів при послідовному з'єднанні виконуються покроково, реалізуючи логічну операцію «**І**». При послідовному підключенні сигнал проходить, коли всі контакти замикаються.

Нормально замкнуті контакти реалізують логічну операцію **АБО** при паралельному підключенні. При паралельному підключенні сигнал проходить, коли один із контактів замкнутий.

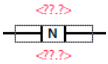


– **Scan operand for positive signal edge** - Елемент «Сканувати операнд для позитивного фронту сигналу», щоб визначити, чи змінюється стан сигналу вказаного операнду (<**Operand1**>) від «0» до

«1». Інструкція порівнює поточний стан сигналу <**Operand1**> із станом сигналу попереднього сканування, який зберігається у заданій комірці пам'яті розміром 1 біт (<**Operand2**>). Якщо інструкція виявляє зміну результату логічної операції (**RLO**) з «0» на «1», є позитивний, висхідний край.

Зміна стану сигналу фіксується кожного разу, коли виконується інструкція. Коли виявляється фронт сигналу, <**Operand1**> встановлюється на стан сигналу «1» для одного програмного циклу. У всіх інших випадках операнд має стан сигналу «0».

Вкажіть операнд, до якого потрібно здійснити запит (<**Operand1**>), у заповнювачі операнду над інструкцією. Вкажіть біт фронтової пам'яті (<**Operand2**>) у заповнювачі операнду під інструкцією.



– **Scan operand for negative signal edge** - Елемент «Сканувати операнд для спадаючого сигналу», щоб визначити, чи змінюється стан сигналу зазначеного операнду (<**Operand1**>) від «1» до «0». Інструкція порівнює поточний стан сигналу <**Operand1**> із станом сигналу попереднього сканування, який зберігається у заданій комірці пам'яті розміром 1 біт <**Operand2**>. Якщо інструкція виявляє зміну результату логічної операції (**RLO**) з «1» на «0», виникає спад сигналу.

Зміна стану сигналу фіксується кожного разу, коли виконується інструкція. Коли виявляється спад сигналу, <**Operand1**> встановлюється на стан сигналу «1» для одного циклу програми. У всіх інших випадках операнд має стан сигналу «0».

Вкажіть операнд, до якого потрібно здійснити запит (<**Operand1**>), у заповнювачі операнду над інструкцією. Вкажіть біт фронтової пам'яті (<**Operand2**>) у заповнювачі операнду під інструкцією.

На наступному рисунку 1 показано зміну стану сигналу у випадку негативного та позитивного фронту сигналу.

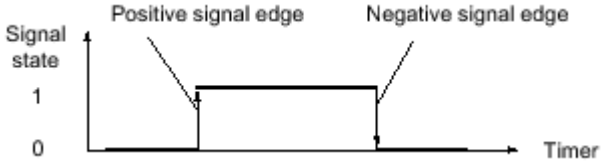
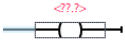
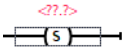


Рис. 1 – Зміна стану сигналу у випадку негативного та позитивного фронту сигналу



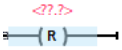
– **Assignment** - Елемент «Призначення», щоб встановити біт зазначеного операнду. Поки результат логічної роботи (**RLO**) на вході елементу має стан сигналу «1», на виході елемента встановлюється стан сигналу «1». Поки стан сигналу «0» на вході елементу, біт зазначеного операнду встановлюється на «0».

Інструкція не впливає на **RLO**. **RLO** на вході елементу направляється безпосередньо на вихід.



– **Set output** - Елемент «Задіяти вихід», щоб встановити стан сигналу вказаного операнду на «1».

Інструкція виконується лише в тому випадку, якщо результат логічної операції (**RLO**) на вході елемента дорівнює «1». Якщо потужність надходить на елемент (**RLO**=«1»), вказаний операнд встановлюється на «1». Якщо **RLO** на вході котушки дорівнює «0» (потік сигналу до елементу відсутній), стан сигналу зазначеного операнду залишається незмінним.



– **Reset output** - Елемент «Відключити вихід», щоб скинути стан сигналу вказаного операнду на «0».

Інструкція виконується лише в тому випадку, якщо результат логічної операції (**RLO**) на вході елемента дорівнює «1». Якщо потужність надходить до елемента (**RLO**=«1»), вказаний операнд скидається до «0». Якщо **RLO** на вході елемента дорівнює «0» (потік сигналу до котушки відсутній), стан сигналу зазначеного операнду залишається незмінним.

FBD (**Function Block Diagram**) - функціональні блокові діаграми. Цей редактор відображає програму у вигляді звичайних логічних схем. Контактів немає, але є еквівалентні функціональні блоки. В даному редакторі не використовується поняття «потік сигналу», як в **LAD**, його висловлює аналогічне поняття потоку управління через логічні блоки **FBD**.

Потоком сигналу називається прохід стану «1» через елементи **FBD**. Логіка програми випливає зі зв'язків між функціональними блоками, що позначають команди.

Графічне представлення функціонального плану добре відображає процес виконання програми.

У лінійній програмі всі команди програми виконуються послідовно одна за одною і знаходяться в одному кодовому блоці,

званому організаційним блоком (**OB1**).

Модульна програма викликає спеціальні кодові блоки, які виконують окремі підзадачі загальної задачі керування. Для створення модульної структури складне завдання автоматизації ділиться на більш прості підзадачі, відповідні технологічні функції процесу.

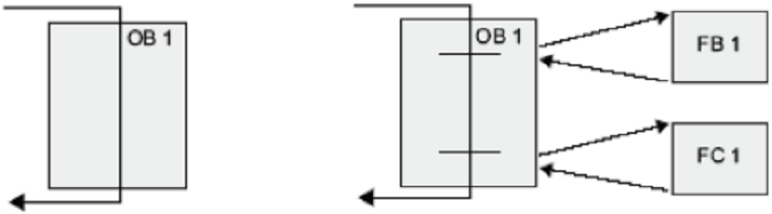


Рис. 2 - Лінійна та модульна структури програми користувача

В ПЛК **Simatic S7-1200** є три типи кодових блоків:

1) Організаційний блок (**OB**) реагує на певні події в **CPU** та може перервати виконання програми користувача. Стандартний блок для виконання програми користувача (**OB1**) представляє основну структуру користувальницької програми і є єдиним кодовим блоком, необхідним для користувача програми. Якщо ви вставите інші **OB** в свою програму, то ці **OB** переривають виконання **OB1**. Інші **OB** виконують специфічні функції, наприклад, для завдань запуску, для обробки переривань і помилок або для виконання конкретного програмного коду через певні інтервали часу.

2) Функціональний блок (**FB**) - це підпрограма, яка виконується при виклику з іншого кодового блоку (**OB**, **FB** або **FC**). Викликаний блок передає параметри в **FB**, а також призначає деякий блок даних (**DB**), який зберігає дані для цього виклику або примірника цього **FB**. Зміна екземпляра **DB** дозволяє родовому **FB** управляти роботою групи пристроїв. Наприклад, один **FB** може керувати декількома насосами або вентилями за допомогою різних екземплярів **DB**, що містять конкретні робочі параметри для кожного насоса або вентиля.

3) Функція (**FC**) - це підпрограма, яка здійснюється за виклику з іншого кодового блоку (**OB**, **FB** або **FC**). У **FC** немає пов'язаного з нею кодового блоку даних **DB**. Викликаючий блок передає параметри в **FC**. Вихідні значення **FC** повинні бути записані в адреси пам'яті або в глобальний **DB**.

Step7-SCL (**Structured Control Language** – структурна керуюча

мова) - це текстова мова високого рівня, розроблена компанією **Siemens** для програмування контролерів серії **Simatic**. Мова **SCL** є текстовою паскалеподібною мовою програмування високого рівня і відповідає в міжнародному стандарті МЕК 61131-3 текстовій мові **ST**.

Для мови **SCL** визначені наступні загальні правила синтаксису:

- одна команда може бути введена в кілька рядків;
- кожна команда завершується символом «;» (крапка з комою);
- немає різниці між малими та великими літерами;
- коментарі в програмі служать тільки для її пояснення та не впливають на її виконання.

Мова **SCL** включає наступні види команд:

- команди присвоєння значень (**Value assignments**) - привласнюють змінній деяке значення (явно вказане значення - константу, результат деякого виразу або значення іншої змінної);
- команди контролю послідовності виконання програми (**Instructions for program control**) - для організації умовних переходів, розгалужень (**IF ... THEN ... ELSE ...**), циклів (**FOR ... TO ... DO ...**) і т. д.;
- інші команди - арифметичні, логічні операції, операції перетворення і округлення, переміщення блоків даних і ін., представлені на панелі **Instructions**;
- команди виклику програмних блоків (**block calls**) - таймерів, лічильників, ПІД-регуляторів і т. д.

IF ... THEN ... ELSE ... - умовою є вираз із булевим значенням (**TRUE** або **FALSE**). Логічний вираз або порівняльний вираз можна вказати як умови.

Коли інструкція виконується, заявлені вирази обчислюються. Якщо значення виразу **TRUE**, умова виконується; якщо значення **FALSE**, воно не виконується.

Залежно від типу гілки, ви можете запрограмувати такі форми інструкції:

Розгалуження через **IF**:

```
IF <умова> THEN <інструкції>
END_IF;
```

Якщо умова виконана, виконуються інструкції, запрограмовані після **THEN**. Якщо умова не виконана, виконання програми продовжується із наступною інструкцією після **END_IF**.

Розгалуження через **IF** та **ELSE**:

IF <умова> THEN <інструкції1>
ELSE <Інструкції0>
END_IF;

Якщо умова виконана, виконуються інструкції, запрограмовані після **THEN**. Якщо умова не виконується, виконуються інструкції, запрограмовані після **ELSE**. Потім виконання програми продовжується із наступною інструкцією після **END_IF**.

Розгалуження через **IF**, **ELSIF** та **ELSE**:

IF <умова1> THEN <інструкції1>
ELSIF <умова2> THEN <інструкція2>
ELSE <Інструкції0>
END_IF;

Якщо перша умова (<**Condition1**>) виконана, виконуються інструкції (<**Instructions1**>) після **THEN**. Після виконання вказівок виконання програми триває після **END_IF**.

Якщо перша умова не виконується, перевіряється друга умова (<**Condition2**>). Якщо виконана друга умова (<**Condition2**>), виконуються інструкції (<**Instructions2**>) після **THEN**. Після виконання інструкцій виконання програми триває після **END_IF**.

Якщо жодна з умов не виконується, виконуються інструкції (<**Instructions0**>) після **ELSE** з подальшим виконанням програми після **END_IF**.

Ви можете вкласти стільки комбінацій **ELSIF** і **THEN**, скільки завгодно, в інструкції **IF**. Програмування гілки **ELSE** є необов'язковим.

CASE – Структура «Створити багатоступінчасту гілку» виконує одну з декількох послідовностей команд залежно від значення числового виразу.

Значення виразу має бути цілим числом. Коли інструкція виконується, значення виразу порівнюється зі значеннями кількох констант. Якщо значення виразу узгоджується зі значенням константи, виконуються інструкції, запрограмовані безпосередньо після цієї константи. Константи можуть приймати такі значення:

- ціле число (наприклад, 5);
- діапазон цілих чисел (наприклад, 15 ... 20);
- перелік, що складається з цілих чи діапазонів (наприклад, 10, 11, 15 ... 20)

Ви можете оголосити інструкцію наступним чином:

CASE <Tag> OF

<Constant1>: <Instructions1>;
<Constant2>: <Instructions2>;
<ConstantX>: <InstructionsX>; // X> = 3
Інакше <Інструкції0>;
END_CASE;

Якщо значення виразу узгоджується зі значенням першої константи (<**Constant1**>), виконуються інструкції (<**Instructions1**>), які запрограмовані безпосередньо після першої константи. Виконання програми згодом поновлюється після **END_CASE**.

Якщо значення виразу не узгоджується зі значенням першої константи (<**Constant1**>), це значення порівнюється зі значенням константи, яка запрограмована наступним чином. Таким чином, інструкція **CASE** виконується до збігу значень. Якщо значення виразу не відповідає жодному із запрограмованих константних значень, виконуються інструкції (<**Instructions0**>), які запрограмовані після **ELSE**. **ELSE** є необов'язковою частиною синтаксису і може бути опущений.

Інструкція **CASE** також може бути вкладена, замінивши блок інструкцій на **CASE**. **END_CASE** являє собою кінець інструкції **CASE**.

Поняття змінної, адресація змінних

Змінна - це область пам'яті контролера для зберігання даних певного типу. Кожна змінна характеризується своєю адресою та довжиною, яка залежить від типу даних, що зберігаються. При зверненні в програмі до змінної для зчитування або запису даних потрібно вказати адресу цієї змінної. У мовах програмування високого рівня є два способи вказівки адреси змінної: абсолютна адреса і символьна адреса. При вказівці абсолютної адреси змінної потрібно вказати область пам'яті контролера, в якій вона знаходиться, розмір змінної, номер початкового байта, і при необхідності - номер біта.

Наприклад, **I0.1**, **QB0**, **MW20**. Перша літера визначає область пам'яті контролера. Друга літера визначає розмір змінної в байтах: **B** (**byte**) - 1 байт, **W** (**word**) - 2 байта, **D** (**double word**) - 4 байта. Однак такий спосіб вказівки адреси не завжди зручний і при використанні в програмі великої кількості змінних може призвести до плутанини при написанні програми. Тому можна використовувати символьну вказівку адреси, при якій з абсолютною адресою змінної зв'язується деяке слово (ім'я змінної), що складається з символів та має для програміста певний зрозумілий йому сенс. Так, наприклад, якщо до дискретного входу контролера **I0.1** підключена кнопка пуску двигуна, то за адресою **I0.1** можна зв'язати символьне ім'я даної змінної **ButtonStart**. Для однієї

змінної може бути поставлено лише одне символьне ім'я.

Для завдання символьних імен змінних в **Simatic Step 7** передбачені таблиці символьних імен - **PLC tags**. Для відкриття таблиці символьних імен необхідно в дереві проекту (**Project tree**) вибрати пункт **PLC_1 → PLC tags → Default tag table**, в результаті чого відкриється таблиця **Default tag table**, створювана середовищем **Simatic Step 7** автоматично для кожного проекту. Для визначення символьного імені змінної необхідно задати саме символьне ім'я (**Name**), вказати її тип даних (**Data type**), абсолютний адресу (**Adress**) і при необхідності - коментар.

ЦПУ надає кілька можливостей для збереження даних під час виконання програми користувача.

Глобальна пам'ять: ЦПУ надає ряд спеціалізованих областей пам'яті, включаючи входи (**I**), виходи (**Q**) і бітову пам'ять (Меркера) (**M**). Ця пам'ять доступна для всіх кодових блоків без обмеження.

M-пам'ять: Будь-який **OB**, **FB** і будь-яка **FC** може звернутися до даних в М-пам'яті, тобто дані знаходяться глобально в розпорядженні всіх елементів програми користувача.

Тимчасова пам'ять: доступ до даних в тимчасовій пам'яті обмежений тим **OB**, **FB** або тієї **FC**, де були створені або оголошені адреси в тимчасовій пам'яті. Адреси тимчасової пам'яті залишаються локальними і не можуть бути використані іншими кодовими блоками, навіть якщо кодовий блок викликає інший кодовий блок. Наприклад: Якщо **OB** викликає **FC**, то **FC** не може звернутися до тимчасової пам'яті **OB**, що викликав цю функцію.

До тимчасової пам'яті можна звертатися тільки з використанням символічної адресації.

Для візуалізації процесів вимірювання, регулювання різних параметрів в промисловості використовують сенсорні панелі. Перевагою використання цих пристроїв є їх компактність, простота програмування, невисока вартість.

Сенсорні панелі різних виробників та різних поколінь володіють різними функціональними можливостями.

Типова панель надає користувачеві наступну функціональність:

- візуалізація параметрів технологічного процесу (або об'єкту) в текстовому або графічному режимах;

- керування та обробка аварійних повідомлень, реєстрація часу і дати виникнення аварійних повідомлень;

- ручне керування за допомогою функціональних кнопок або сенсорного екрану;

- можливість програмування графіки і налаштування функціональних клавіш;

- побудова діаграм та трендів, відображення зведених звітів.

Тензодатчик **Siemens SIWAREX WL 260** серії **SP-S AA**

Датчики сило-вимірювальні тензорезисторні ДСТ Датчики сило-вимірювальні, з клейовими фольгових тензорезисторами і з'єднаними по мостовій схемі, призначені для роботи в сило-вимірювальних системах, з метою вимірювання статичних або повільно змінюваних зусиль (стиснення і/або розтягування) (рис. 3).

Рис. 3 – Загальний вигляд датчика Siemens SIWAREX WL 260 серії SP-S AA

Зміна опору є дуже малою (зазвичай менш 0,1 Ом при номінальному опорі 100 Ом) і за величиною порівняна зі зміною під впливом температури. Тому зазвичай на навантажувального осередку закріплюються чотири тензодатчика, з яких два сприймають навантаження (деформуються), а два, розташовані під кутом 90°, є ненавантаженими.

Включення цих тензодатчиків в міст Уітстона (рис. 4) дозволяє отримати вихідний сигнал, що залежить тільки від навантаження і не залежить від температури.

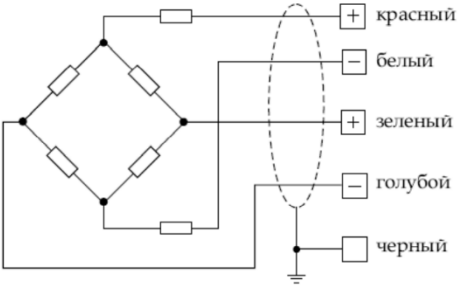


Рис.4 - Схема електрична принципова тензорезисторного датчика

При підключенні тензодатчиків до уніфікованих аналоговим входам контролера необхідно використовувати проміжний підсилювач (рис. 5).

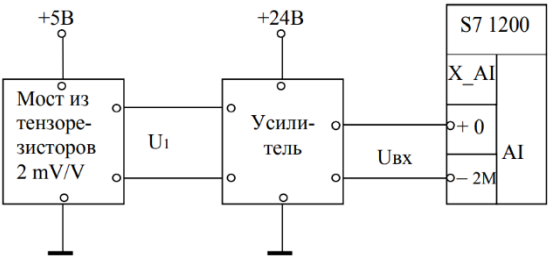


Рис. 5 – Схема електрична проміжного підсилювача

Порядок виконання лабораторної роботи

В ході виконання лабораторної роботи необхідно створити в середовищі **TIA Portal** новий проект, використовуючи вже існуючі підпрограми по запуску маніпулятора (рис.6), конвеєра (рис.7) та зважувальної дільниці (рис.8) в якому реалізувати алгоритм автоматичного керування процесу переміщення продукції. Створити візуалізацію з використанням можливостей **HMI**-інтерфейсу.

Рис. 6 – Загальний вигляд маніпулятора

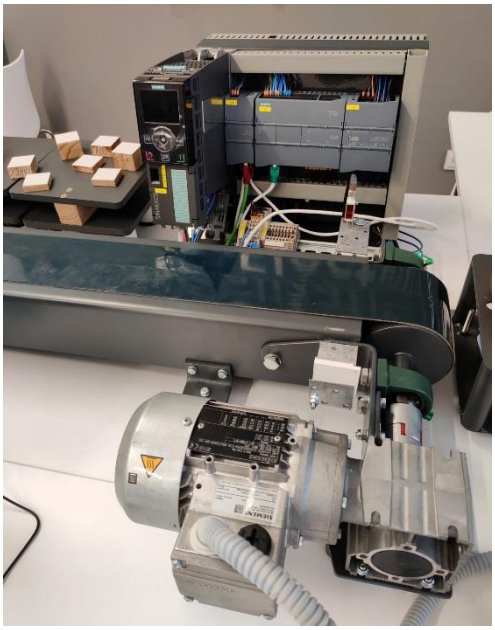


Рис. 7 – Загальний вигляд конвеєра та частотного перетворювача

Рис.8 – Загальний вигляд зважувальної дільниці

Зв’язати роботу зважувальної дільниці із автоматичним запуском маніпулятора в роботу, створити за допомогою структури **CASE** умову для вмикання маніпулятора.

Використовуючи функції розгалуження **IF ... THEN ... ELSE**

розписати роботу маніпулятора по переміщенню продукції між позиціями з відслідковуванням точного положення продукції та наступним встановленням на конвеєр. Створити в програмі керування маніпулятора можливість переривання процесу автоматичного переміщення у випадку виникнення помилок.

Автоматичний запуск конвеєра виконати з використанням можливостей мови релейних діаграм та оптичних датчиків.

Завантажити створену програму в контролер, досліджувати і продемонструвати викладачеві її роботу на лабораторному стенді.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд HMI-панелі та програми процесу автоматизованого переміщення продукції.
3. Повний опис побудови програми автоматизованого переміщення продукції.
4. Висновки з роботи.

Контрольні питання

1. Як створити проект в середовищі програмування **TIA** (**Totally Integrated Automation**) **Portal** (**V15**)?
2. Релейні діаграми, основні елементи.
3. Функціональні блокові діаграми.
4. Типи кодових блоків.
5. Структурна керуюча мова.
6. Який принцип дії розгалуження **IF ... THEN ... ELSE ...**?
7. Структура **CASE.**
8. Поняття змінної, адресація змінних
9. Будова та принцип дії тензодатчика.
10. Чому в тензодатчику використовуються 4 тензорезистора?

Перелік джерел посилань

1. Программируемый контроллер S7-1200. Системное руководство. 11/2009, A5E02669003-02.
2. Куцик А. Автоматизовані системи керування на програмованих логічних контролерах: навч. посіб. / А. Куцик, В. Місюренко. – Львів: Видавництво Львівської політехніки, 2011. – 200 с.
3. Пупена О.М. Промислові мережі та інтеграційні технології в автоматизованих системах : навч. посіб / [Пупена О.М., Ельперін І.В., Луцька Н.М., Ладанюк А.П.]. – К. : Вид-во «Ліра-К», 2011. – 552 с.