Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризированих технологій
(повна назва)
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації
(повна назва)
та робототехніки

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)
(рівень вищої освіти)

Розробка програмного забезпечення для систем автоматизації для конвеєрної лінії на основі мови релейної логіки LAD
(тема)

Виконав:
Студент 3(прискореного) курсу, групи
АКТСІу-21-1
Глєбов Євген Олександрович
(прізвище, ім'я, по батькові)
Спеціальності 151 Автоматизація та
комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
Освітня програма Системна інженерія
(назва)
Керівник проф. кафедри КІТАР Цимбал О.М.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. Кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. Кафедри КІТАР
(підпис)
«____» 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Глєбову Євгену Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для систем автоматизації конвеєрною лінією на основі мови релейної логіки LAD
Затверджена наказом університету від 20.05.2024 р. № 478 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 2024 р
3. Вихідні дані до роботи:
3.1 Siemens S7-1214DC, Siemens S7- 1511DC;
3.2 Оптичні датчики:
3.3 Середовище розробки TIA Portal;
3.3 Мова програмування LAD,FBD,SLC;
3.4 Можливість апаратної реалізації
4. Перелік питань, що потрібно опрацювати в роботі:
4.1 Вступ;
4.2 Визначення мети , об'єкту та предмету розробки;
4.3 Аналіз предметної області;
4.4 Розробка топології та вибір обладнання для конвеєрної лінії;
4.5 Розробка програмного забезпечення автоматичної конвеєрної системи на основі мови релєйної логіки LAD
4.6 Висновки.

5. Графічний демонстраційний матеріал в форматі powerpoint(*.ppt) формату А4 –15 сторінок.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		Підпис	Дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Актуальність роботи, постановка задачі	08.04.24 – 16.04.24	виконано
2	Визначення мети об’єкту і предмету розробки	16.04.24 – 27.04.24	виконано
3	Аналіз предметної області	01.05.24 – 04.05.24	виконано
4	Розробка топології та вибір обладнання для конвеєрної лінії	04.05.24 – 12.05.24	виконано
5	Розробка програмного забезпечення автоматичної конвеєрної системи а основі мови релейної логіки LAD	12.05.24 – 28.05.24	виконано
6	Подання роботи на перевірку Інтернет-сервісом Unicheck	28.05.24 – 04.06.24	виконано
7	Оформлення пояснювальної записки	26.05.24 – 08.06.24	виконано
8	Подання роботи на рецензію	08.06.24 – 09.06.24	виконано
9	Подання роботи на підпис зав. Кафедри	09.06.24 – 10.06.24	виконано
11	Подання кваліфікаційної роботи в ЕК	10.06.24 – 11.06.24	виконано

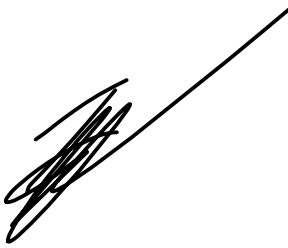
Дата видачі завдання 08.04 2024 р.

Студент _____ Глєбов Є.О.
(підпис) (прізвище, ініціали)

Керівник роботи професор Цимбал О.М.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«11» червня 2024 р. Глєбов Є.



РЕФЕРАТ

Пояснювальна записка: 91 с., 31 рис., 1 дод., 11 джерел.

СИСТЕМА КЕРУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОНВЕЄРНА ЛІНІЯ, КОНТРОЛЕРИ, ТЕСТУВАННЯ

Об’єкт розробки – процес керування конвеєрною лінією.

Предмет розробки – програмне забезпечення для контролерів Siemens S7-1200 та S7-1500.

Мета дослідження –. створення ефективної та надійної системи автоматизованого керування конвеєрною лінією, що дозволить оптимізувати виробничі процеси та зменшити витрати на експлуатацію обладнання.

В кваліфікаційній роботі проведено аналіз існуючих аналогів та технологічних рішень у сфері автоматизації конвеєрних систем, а також досліджено особливості функціонування контролерів Siemens S7-1200 та S7-1500. Розроблено структурну схему автоматизованої системи та створено програмне забезпечення на мові LAD для керування конвеєрною лінією. Після цього було проведено тестування та оптимізацію розробленої системи.

У ході роботи було створено ефективну систему автоматизованого керування конвеєрною лінією. Розроблене програмне забезпечення забезпечує коректну роботу конвеєрної системи, підвищуючи ефективність виробничих процесів та знижуючи витрати на експлуатацію обладнання. Висновки дослідження показали, що розроблене програмне забезпечення на основі мови LAD для контролерів Siemens S7-1200 та S7-1500 дозволяє оптимізувати роботу конвеєрних ліній, що є важливим кроком у напрямку підвищення ефективності виробничих процесів. Результати роботи можуть бути використані для подальшого розвитку систем автоматизації в різних галузях промисловості.

ABSTRACT

The explanatory note: 91 p., 31 figures, 1 pp, 11 sources.

CONTROL SYSTEM, SOFTWARE, CONVEYOR LINE, CONTROLLERS, TESTING

The object of development is an automated conveyor line control system.

The subject of development is software for controllers Siemens S7-1200 and S7-1500.

The aim of the study – creation of an effective and reliable system of automated control of the conveyor line, which will allow to optimize production processes and reduce equipment operation costs.

In the qualification work, an analysis of existing analogues and technological solutions in the field of automation of conveyor systems was carried out, as well as the peculiarities of the functioning of the Siemens S7-1200 and S7-1500 controllers were investigated. The structural diagram of the automated system was developed and software in the LAD language was created for the control of the conveyor line. After that, the developed system was tested and optimized.

During the work, an effective system of automated control of the conveyor line was created. The developed software ensures the correct operation of the conveyor system, increasing the efficiency of production processes and reducing equipment operation costs. The research findings showed that the developed software based on the LAD language for the Siemens S7-1200 and S7-1500 controllers allows for the optimization of conveyor lines, which is a step towards increasing the efficiency of production processes. The results of the work can be used for the further development of automation systems in various industries.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АСУ – Автоматизована система управління;

АСУТП – Автоматизована система управління технологічним процесом;

АТК – Автоматизований технологічний комплекс;

ІЕДФ0 – IDEF0 діаграма;

ТОУ – технічний об’єкт управління;

ПЛК – Програмований логічний контролер;

AI – Artificial Intelligence;

IDEF0 – Integrated definition for Function Modeling;

LAD – Ladder Logic;

PLC – Programmable Logic Controller;

PN – Part Number;

SICK – Sensor Intelligence by SICK AG;

TIA – Total Integrated Automation.

ВСТУП

Сучасне промислове виробництво неможливе без автоматизації технологічних процесів, що значно підвищує ефективність, точність та продуктивність. Автоматизація конвеєрних систем відіграє ключову роль у багатьох галузях промисловості, забезпечуючи безперервний потік продукції та зменшуючи людський фактор у виробничих процесах. Ця робота присвячена розробці програмного забезпечення для автоматизації конвеєрної лінії на основі використання мови релейної логіки (LAD). Вибір мови зумовлений її широким застосуванням у програмуванні контролерів Siemens серії S7-1200 та S7-1500, які є одними з найбільш поширених у промисловій автоматизації.

Метою роботи є створення ефективної та надійної системи автоматизованого керування конвеєрною лінією, що дозволить оптимізувати виробничі процеси та зменшити витрати на експлуатацію обладнання. Для досягнення цієї мети необхідно вирішити наступні завдання: провести аналіз існуючих аналогів та технологічних рішень у сфері автоматизації конвеєрних систем; дослідити особливості функціонування контролерів Siemens S7-1200 та S7-1500; розробити структурну схему автоматизованої системи; створити програмне забезпечення на мові LAD для керування конвеєрною лінією; провести тестування та оптимізацію розробленої системи.

Об’єктом дослідження є автоматизована система керування конвеєрною лінією, а предметом – програмне забезпечення для контролерів Siemens S7-1200 та S7-1500.

Аналізуючи автоматизацію виробничих процесів, можна прийти до висновку, що ключовим моментом є розробка програми для керування цими системами. Одним із передових виробників таких контролерів є компанія Siemens та її контролери Siemens S7-1200 та S7-1500. Для налаштування та програмування даних контролерів компанія Siemens розробила інтегроване середовище розробки

TIA Portal, яке дозволяє напряму або віддалено займатися налаштуванням та програмуванням логік цих контролерів.

Теоретичною основою автоматизації виробничих процесів є теорія продуктивності, яка дозволяє розглядати як автоматизацію виробничих процесів, так і автоматизацію конвеєрних та інших систем, вирішувати конкретні задачі логістики, розрахунку і вибору технологічних, конструктивних структур, надійності й ефективності. Сучасні конвеєрні системи – це не просто секція з мотором, а технологічна секція з датчиками, моторами та власними контролерами, для керування якими використовуються контролери Siemens. Ще однією важливою рисою сучасної автоматизації є розширення використовуваних технічних засобів, що призводить до більшого розмаїття варіантів для автоматизації виробничих процесів.

Стратегія комплексної автоматизації електронного апаратобудування як основи технічної політики визначається кількома аспектами, у тому числі: правильним розумінням змісту й основної спрямованості робіт з автоматизації; об’єктивною оцінкою в часі перспективності та доцільності застосування нових методів і засобів автоматизації, їхнього сполучення та зв’язку з аналогами. Надзвичайну актуальність у рамках стратегії автоматизації набуває проблема об’єктивної оцінки і розумного впровадження новітніх методів і засобів автоматизації, тому що це пов’язано з великими капітальними витратами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В основній частині систем автоматизації знаходяться компоненти, які задовольняються такими властивостями, як відкритість, стабільність, масштабність, якщо брати відкритість – означає застосування відкритих стандартів, що визначають гнучкість архітектурних систем автоматизації, можливість взаємодії з іншими системами [5].

Перш ніж починати програмування контролерів, спочатку потрібно зрозуміти алгоритм виробництва та задачі які мають виконуватися для досягнення цілей. Для цього було створено контекстну (рис 1.1).

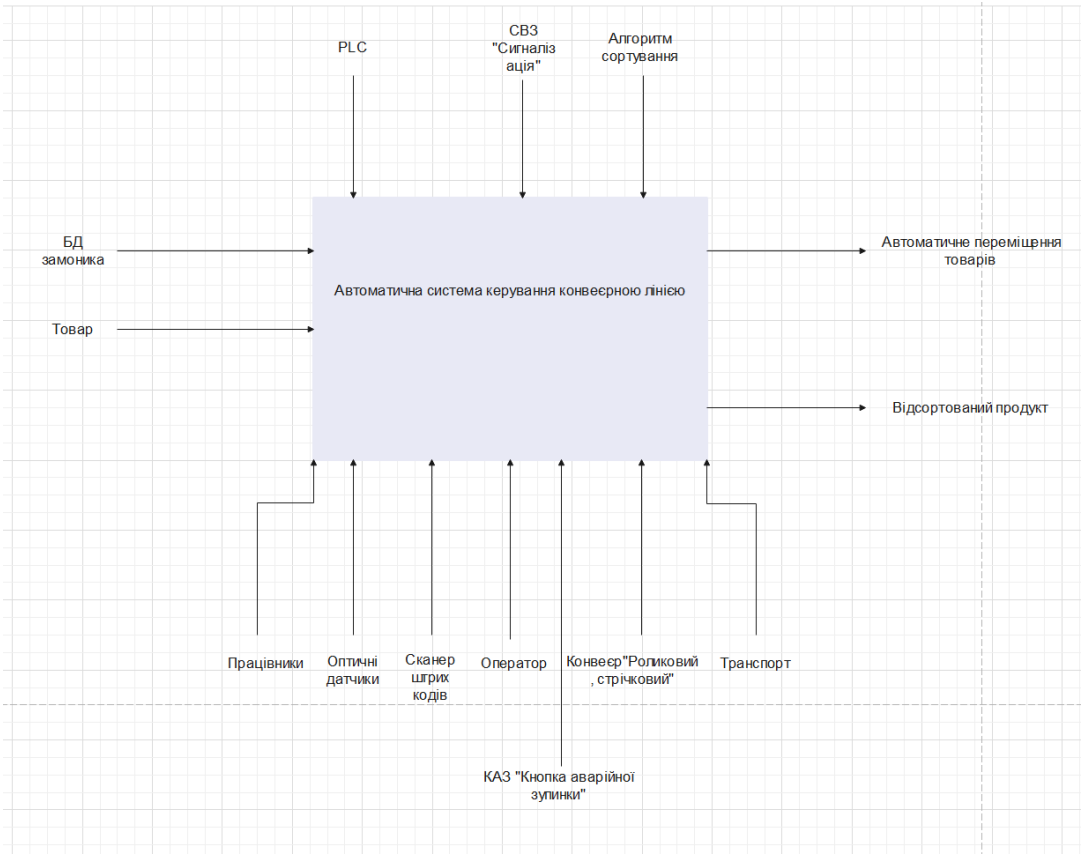


Рисунок 1.1 – Контекстна діаграма автоматичної системи керування конвеєрною лінією

Ознайомившись з алгоритмом роботи ми можемо проаналізувати всі необідні критерії для керування АСУ конвеєрної системи. Та запрограмувати контролери заданим умовам.

Критерії управління автоматизованими системами управління технологічними процесами (АСУТП) включають техніко-економічні показники, такі як собівартість готового продукту та продуктивність технічного об'єкта управління (ТОУ), зберігаючи при цьому стандартну якість продукції. Інші критерії — це технологічні показники, наприклад, параметри технологічного процесу або якості кінцевого продукту. Система, що об'єднує АСУТП та ТОУ в одне ціле, називається автоматизованим технологічним комплексом (АТК). Щоб отримати інформацію про поточний стан об'єкта в режимі реального часу, використовуються різноманітні датчики, що вимірюють технологічні параметри, такі як температура, тиск, витрата, якість, а також датчики стану обладнання (наприклад, "ввімкнено" чи "вимкнено").

Основні переваги АСУТП включають зниження або навіть повне усунення впливу людського фактора на керований процес, скорочення кількості персоналу, зменшення витрат на сировину, покращення якості кінцевого продукту, і в підсумку значне підвищення ефективності виробництва. Серед ключових функцій таких систем - контроль і управління, обмін даними, обробка, накопичення та зберігання інформації, створення тривожних сигналів, а також побудова графіків та звітів.

З розвитком обчислювальної техніки та вдосконаленням методів управління виробництвом, зростає увага до автоматизованих систем, що керують замкненими виробничими ділянками та процесами автоматизації. Його основні напрями:

– автоматичне регулювання (стабілізація);

– програмне управління;

– системи, що стежать;

– екстремальне регулювання;

– оптимальне управління;

– самонастроювальні системи.

Процес управління, у загальному випадку, представляє процес обробки й переробки інформації. Таким чином, функціональний склад суб'єкта управління може бути представлений наступними підсистемами (блоками):

– блок прийняття рішень;

– блок програмно-математичного забезпечення;

– блок технічного забезпечення (блок технічних засобів).

Основним елементом системи є блок прийняття рішень, що складається з групи фахівців, організованих у вигляді ієрархічної структури, яка застосовує різні методи для прийняття рішень. Ефективна робота цього блоку можлива лише за умови комплексного підходу до розробки і реалізації проекту з автоматизації підприємства.

Процес розробки проекту можна звести до наступної послідовності:

– дослідження функцій підприємства і його підрозділів;

– розробка функціональної моделі діяльності організації;

– розробка інформаційної моделі;

– складання моделі полій організації;

– розробка пропозицій по оптимізації.

При розробці системи враховується, що на якість функціонування системи управління будуть впливати:

– рівень і характер інформаційного забезпечення процесів управління;

– умови збору й обробки інформації для аналізу та синтезу процесів;

– якість, вірогідність і повнота інформації (зовнішньої та внутрішньої);

– стандартизація й ідентифікація вихідних даних;

– імовірнісні характеристики одержуваної інформації як по об'єкту, так і по межах;

– ступінь досконалості математичних моделей підсистем і системи в цілому;

Основні функції систем АСУ ТП Під функцією системи зазвичай розуміють деяку сукупність дій, спрямованих на досягнення тієї чи іншої цілі управління, а це у свою чергу визначається особливостями і вимогами об'єкта (рис. 1.3). Усе

різноманіття функцій, що виконуються АСУ ТП, прийнято ділити на:

– управляючі;

– інформаційні;

– допоміжні.

Інформаційні функції автоматизованої системи управління замкненою виробничою ділянкою включають збір, обробку та надання інформації про стан технологічного об'єкта оператору або для подальшої обробки в блокі формування керуючих впливів. У процесі обробки інформації виконуються операції, як-от підсумовування, згладжування, обчислення непрямих показників, які не можуть бути визначені безпосередньо під час контролю, і порівняння поточних значень параметрів технологічного процесу із заданими. Одночасно може відбуватися підготовка і передача інформації в інші суміжні системи управління, узагальнення результатів, а також прогнозування стану технологічного об'єкта та обладнання.

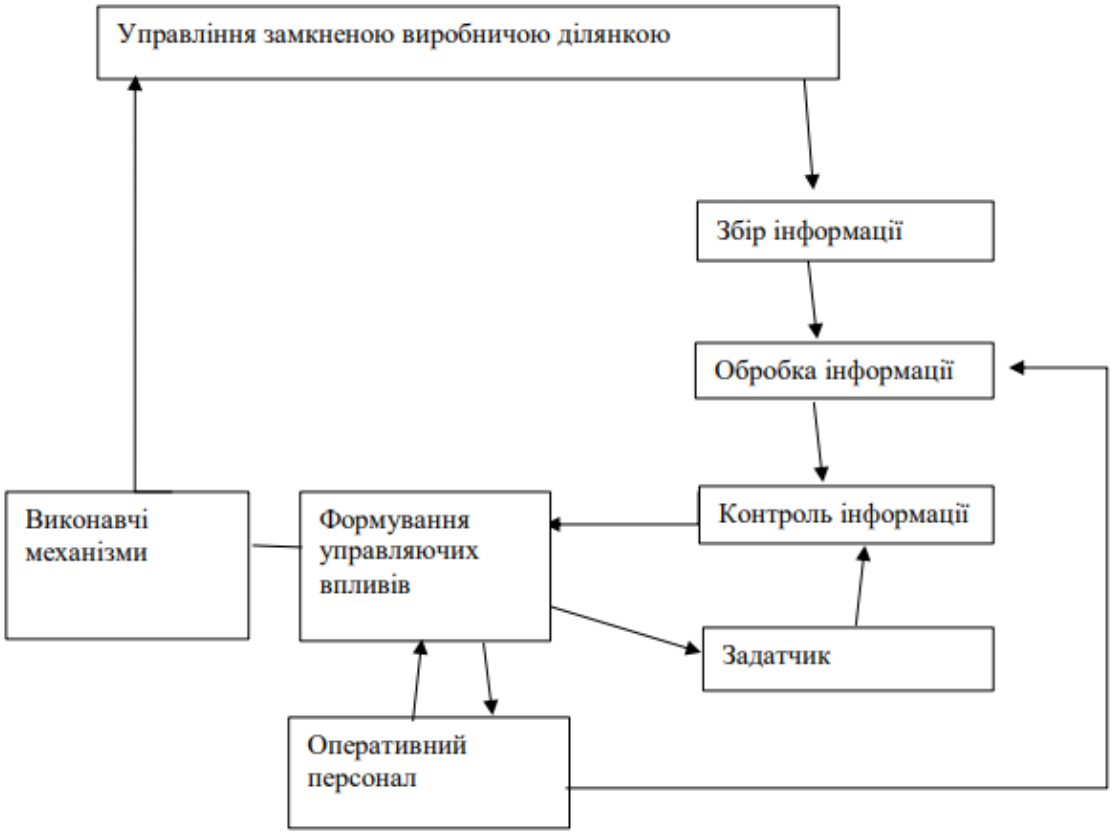


Рисунок 1.3– Узагальнена функціональна структура автоматизованого управління замкненою виробничою ділянкою

Відмінною рисою керуючих та інформаційних функцій АСУ ТП є їхня орієнтованість на конкретного користувача. Допоміжні функції забезпечують вирішення внутрішніх системних завдань. На відміну від керуючих та інформаційних функцій, допоміжні функції призначені для забезпечення функціонування самої автоматизованої системи. До них відноситься самоконтроль окремих компонентів або всієї системи в цілому.Управління технологічним об’єктом:

– збір інформації;

– обробка інформації;

– контроль інформації.

Найбільш розповсюдженими сьогодні є АСУ з інформаційними функціями для забезпечення користувача необхідною інформацією про стан і характеристики об’єкта, що управляється.

У цьому випадку головними задачами:

– збір інформації з окремих точок технологічного об’єкта;

– централізована обробка інформації за алгоритмами системи.

Під централізованою обробкою інформації розуміють порівняння значень окремих параметрів з прийнятими уставками, сигналізацію, а іноді реєстрацію відхилень параметра від норми, вимірювання і (або) реєстрацію по виклику оператора окремих параметрів процесу, обчислення, а при необхідності і реєстрацію ряду комплексних показників, включаючи техніко-економічні, які характеризують хід і ефективність функціонування об’єкта.

Сьогодні важливою вимогою до АСУ ТП є вирішення ситуаційних задач, які забезпечують оператору можливість отримання автоматично або за викликом нової картини виробничої ситуації на тій чи іншій дільниці процесу.

2 РОЗРОБКА ТОПОЛОГІЇ ТА ВИБІР ОБЛАДНАННЯ ДЛЯ КОНВЕЄРНОЇ ЛІНІЇ

2.1 Контролери для керування системою

Ключовою, або навіть можна сказати головною частиною будь якої автоматизованиї системи є PLC (Програмований логічний контролер (ПЛК)) – це спеціалізований тип комп'ютера, який широко використовується в промисловій автоматизації та системах управління. Він призначений для відстеження вхідних даних, прийняття рішень на основі запрограмованої логіки та керування вихідними сигналами, що дозволяє автоматизувати різні механізми та процеси. ПЛК відомі своєю надійністю, міцністю та здатністю працювати в суворих промислових умовах, що робить їх незамінними для сучасної промисловості.

Ці контролери мають ряд ключових особливостей, які відрізняють їх від звичайних комп'ютерів. ПЛК розроблені для роботи в середовищах, де можливі екстремальні температури, вібрації, пил та електричні перешкоди. Ця витривалість дозволяє їм функціонувати на промислових об'єктах, таких як фабрики, заводи та інші виробничі майданчики, без значних збоїв або необхідності частого обслуговування [3-4].

Одними з найпопулярніших серій ПЛК є Siemens S7-1200 і S7-1500. Кожен із цих контролерів має унікальні переваги, що робить їх підходящими для різних застосувань. Siemens S7-1200 – це компактний ПЛК, який ідеально підходить для невеликих та середніх систем автоматизації зображено на рисунках 2.1-2.2. Він відомий своєю гнучкістю, простотою використання та доступністю, що робить його привабливим для підприємств, які хочуть автоматизувати процеси з мінімальними витратами. Він також має інтегровані функції мережевого підключення, підтримує різні протоколи, такі як Profinet та Modbus, і може програмуватися за допомогою середовища TIA Portal (Totally Integrated Automation). Завдяки своїй гнучкості S7-1200 можна використовувати для

автоматизації невеликих виробничих ліній, машин, а також для систем збору даних.

Рисунок 2.1 – Контролер Siemens S7-1200

З іншого боку, Siemens S7-1500 – це потужніший ПЛК, призначений для складніших систем автоматизації. Він має вищу продуктивність, розширені функції діагностики, більшу кількість модулів вводу/виводу та більшу ємність пам'яті порівняно з S7-1200. Ці характеристики дозволяють S7-1500 справлятися зі складними промисловими системами та великими виробничими лініями. Як і S7-1200, S7-1500 також програмується через TIA Portal, але він також може виконувати складніші завдання і може використовуватися для систем керування процесами та інших масштабних проектів.

Рисунок 2.2 – Контролер Siemens S7-1500

Обидва контролери мають модульну архітектуру, що дозволяє користувачам додавати або видаляти модулі вводу/виводу залежно від потреб проекту. Вони підтримують різні протоколи зв'язку, такі як Profinet, Profibus, Modbus, що дозволяє легко інтегрувати їх в загальні системи автоматизації. Завдяки цьому ці контролери можуть взаємодіяти з іншими пристроями та системами, створюючи комплексні мережі автоматизації.

Вибір між S7-1200 і S7-1500 залежить від специфічних потреб проекту. Якщо потрібно автоматизувати невеликі системи, прості виробничі лінії або машини, S7-1200 – це хороший вибір завдяки своїй компактності та гнучкості. Але якщо потрібна більша потужність, можливість виконувати складніші завдання і більше розширень, то S7-1500 буде кращим вибором.

Загалом, ПЛК серій S7-1200 та S7-1500 від Siemens є важливими інструментами для промислової автоматизації. Вони дозволяють підприємствам підвищити ефективність, зменшити витрати на робочу силу та забезпечити більшу надійність виробничих процесів.

2.2 Інтелектуальний модуль управління конвеєром conveylinx AI2

Conveylinx AI2 – це інтелектуальний модуль управління, розроблений для автоматизації конвеєрних систем і транспортувальних ліній, з можливостями управління, які роблять його надзвичайно гнучким та багатофункціональним. Цей модуль використовується для керування приводами на конвеєрах, дозволяючи забезпечувати оптимальний контроль над рухом, швидкістю та напрямком. Завдяки таким можливостям, conveylinx AI2 широко застосовується в різних галузях промисловості, таких як логістика, складські системи, виробничі лінії, а також системи сортування та пакування зображено на рисунку 2.3.

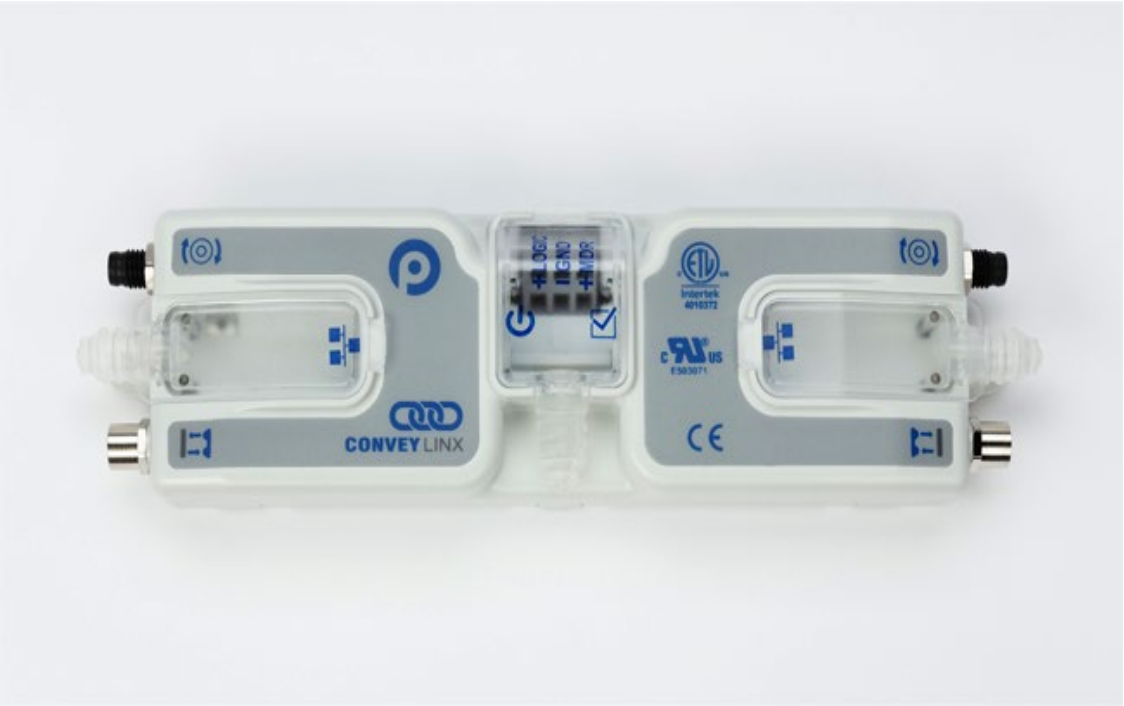


Рисунок 2.3 – Інтелектуальний модуль управління конвеєром conveylinx AI2

Однією з ключових переваг conveylinx AI2 є його модульність і гнучкість. Модуль може контролювати окремі ролики або секції конвеєра, дозволяючи створювати складні конфігурації руху. Це дає змогу розділяти конвеєр на зони, де кожна зона може працювати незалежно одна від одної. Наприклад, у системі із зональною зупинкою conveylinx AI2 може зупиняти ролики в тих місцях, де немає вантажу, що сприяє економії енергії та зменшенню зношування обладнання.

Conveylinx AI2 також підтримує різні протоколи комунікації (рис 2.4), такі як Ethernet/IP, Modbus TCP/IP та інші, що дозволяє легко інтегрувати його в сучасні системи автоматизації. Ця здатність до інтеграції робить його сумісним із різними системами контролю та управління, включаючи системи збору даних, SCADA-системи, а також роботизовані системи [11]. Це дозволяє створювати комплексні автоматизовані рішення, що поєднують різні технології

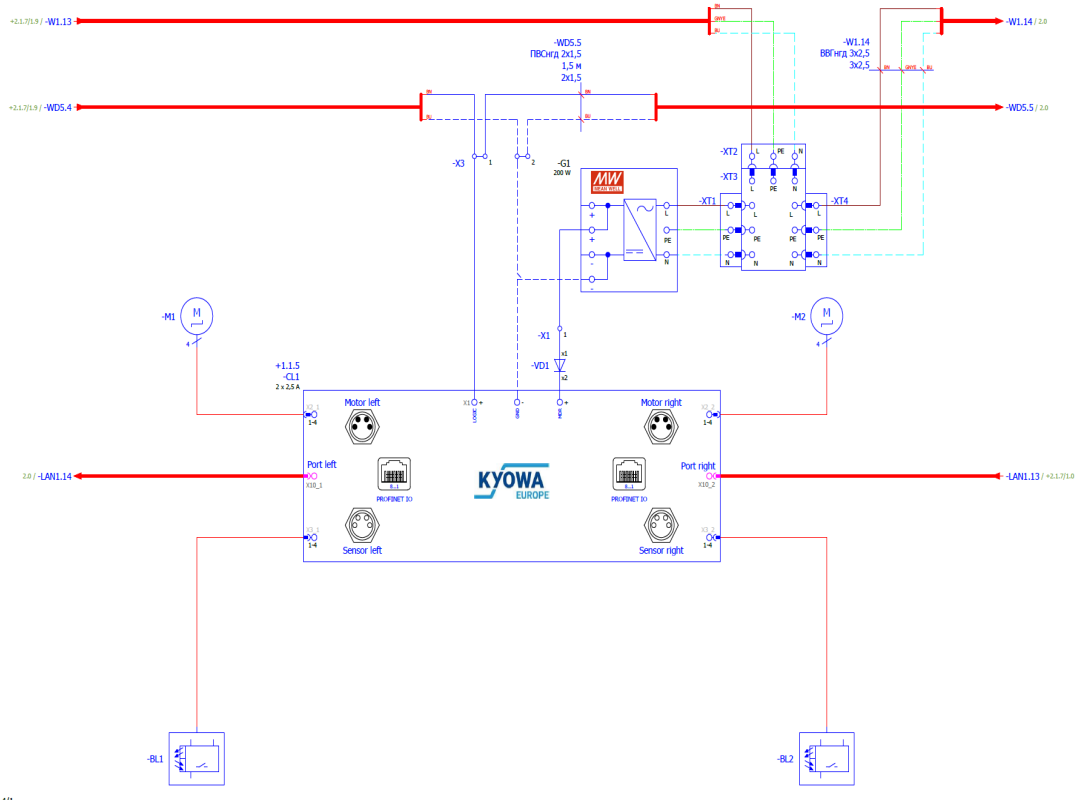


Рисунок 2.4 – Схема підключення conveylinx AI2 на конвеєрній системи.

Ще однією перевагою conveylinx AI2 є його ефективність у використанні енергії. Завдяки функціям оптимізації, модуль дозволяє контролювати приводи таким чином, щоб мінімізувати енергоспоживання, коли вони не використовуються. Це не тільки знижує витрати на енергію, але й сприяє подовженню терміну служби обладнання. Крім того, conveylinx AI2 забезпечує додаткові функції безпеки, такі як автоматичне зупинення у випадку несправності або перевантаження, що забезпечує безпеку персоналу та обладнання.

Як приклад аналогів conveylinx AI2, можна згадати модулі керування від

компаній Interroll та Itoh Denki, які також пропонують системи автоматизації для конвеєрів. Однак conveylinx AI2 виділяється своєю здатністю до гнучкого програмування та широкими можливостями налаштування. Він дозволяє контролювати не тільки швидкість і напрямок руху, але й встановлювати складні графіки роботи, що робить його ідеальним для використання в системах із змінним навантаженням.

Керування conveylinx AI2 здійснюється за допомогою спеціалізованого програмного забезпечення, яке має інтуїтивний інтерфейс і дозволяє здійснювати налаштування безпосередньо з комп'ютера або іншого пристрою. Це програмне забезпечення надає можливість відстежувати стан модуля в реальному часі, проводити діагностику та швидко змінювати параметри за потреби. Такий підхід до управління сприяє підвищенню ефективності системи та полегшує технічне обслуговування.

Таким чином, conveylinx AI2 є потужним інструментом для автоматизації конвеєрів, який забезпечує високу гнучкість, енергоефективність та інтеграцію з іншими системами. Це робить його оптимальним вибором для сучасних виробничих та логістичних систем, де необхідний точний контроль, безпека та надійність.

2.3 Оптичні датчики SICK

SICK – це німецька компанія, яка спеціалізується на виробництві високоякісних сенсорних систем для промислового використання. Вона відома своїм широким асортиментом продуктів, включаючи датчики наближення, фотоелектричні датчики, лазерні сканери, датчики переміщення та інші. SICK має добре заслужену репутацію у галузі промислової автоматизації та логістики.

Одним з найпопулярніших продуктів SICK є фотоелектричні датчики. Вони працюють на основі оптичних принципів, що дозволяє виявляти об'єкти без фізичного контакту. В основі їх роботи лежить випромінювання світлового променя та детекція відбиття цього променя від об'єкта. Якщо відбиття

зареєстроване, це означає, що об'єкт виявлений. Фотоелектричні датчики від SICK відрізняються високою точністю та швидкістю реакції, що робить їх ідеальними для застосувань, де потрібна висока продуктивність.

Переваги цих датчиків включають високу надійність, гнучкість в налаштуванні, а також здатність працювати на великих відстанях. Вони також здатні працювати в складних умовах, таких як високі температури, вологість та запиленість. Це робить їх придатними для використання в різноманітних галузях, від виробничих ліній до систем безпеки.

Недоліки можуть виникати у випадках, коли світло відображається або розсіюється іншими об'єктами, що може викликати помилкові спрацьовування. Також іноді потрібно додаткове калібрування та налаштування, щоб забезпечити точність роботи датчика. Ще однією потенційною проблемою є висока ціна порівняно з іншими типами датчиків, що може бути суттєвим фактором для деяких користувачів.

Як аналоги фотоелектричних датчиків можна розглядати інші типи сенсорів, наприклад, ультразвукові та індуктивні датчики. Ультразвукові датчики використовують звукові хвилі для виявлення об'єктів і можуть бути корисними в середовищах, де світлові умови непередбачувані або дуже складні. Індуктивні датчики добре працюють з металевими об'єктами, використовуючи електромагнітні поля для виявлення.

SICK також пропонує широкий спектр інших датчиків, таких як лазерні сканери, що використовуються в системах безпеки та контролю, датчики переміщення для вимірювання позицій та відстані, а також RFID-датчики для ідентифікації об'єктів. Ці датчики мають різні характеристики і використовуються у різноманітних галузях: від автомобільної промисловості до харчової та фармацевтичної зображено на рисунку 2.5.

Рисунок 2.5 – Варіація датчиків SICK.

Характеристики датчиків SICK можуть змінюватись залежно від моделі та застосування, але загалом вони відомі своєю точністю, надійністю та довговічністю. SICK продовжує інвестувати в розробку нових технологій та підвищення якості своєї продукції, що робить її одним із провідних виробників сенсорів у світі.

Характеристики обраного для виробництва датчика:

Назва датчика: WTT12LC-B2543.

Тип датчика: Фотоелектричний датчик (датчик відстані)

Характеристики:

– принцип роботи: технологія Time-of-Flight (tof), яка дозволяє вимірювати відстань за допомогою світлового імпульсу;

– діапазон вимірювання: від 50 до 1 000 мм;

– точність вимірювання: в межах ±10 мм;

– час реакції: 3 мс;

– вихідний сигнал: PNP/NPN, залежно від налаштувань;

– електроживлення: 10-30 В DC;

– ступінь захисту: IP67, що забезпечує захист від пилу та вологи;

– матеріал корпусу: пластик ABS;

– робоча температура: від -40 до +60 градусів Цельсія.

Цей датчик використовується в автоматизованих системах для вимірювання відстані, виявлення об'єктів і позиціювання. Завдяки технології tof він забезпечує високу точність і стабільність у роботі навіть за несприятливих умов. Його діапазон вимірювання дозволяє застосовувати його в різноманітних галузях, таких

як логістика, виробництво та системи контролю якості. Завдяки корпусу з високим ступенем захисту, цей датчик може використовуватися в складних умовах, де є ризик пилу або вологи (рис 2.6).

Рисунок 2.6 – Встановлення датчику на конвеєрній лініій

2.4 Сканера штрих-кодів SICK CLV62x

Стаціонарні сканери штрихкода SICK CLV62x є популярним вибором для автоматизації процесів, що пов'язані зі скануванням штрихкодів. Ці сканери використовуються в різних галузях, включаючи виробництво, логістику, складське зберігання та роздрібну торгівлю. Вони пропонують надійне рішення для автоматизованого зчитування штрихкодів, забезпечуючи високу точність і швидкість (рис 2.7).

Рисунок 2.7 – Сканера штрих-кодів SICK CLV62x

Переваги сканерів SICK CLV62x включають відмінну продуктивність сканування навіть при поганій якості штрихкодів. Завдяки потужному лазеру та ефективним алгоритмам обробки, ці сканери можуть зчитувати штрихкоди з різних матеріалів, включаючи папір, пластик і метал. Вони також здатні працювати на великій відстані, що дозволяє їх використовувати в ситуаціях, де доступ до об'єкта сканування обмежений.

Ще одна перевага SICK CLV62x – це їхня гнучкість у налаштуванні. Вони можуть бути під'єднані до різних систем автоматизації та контролю завдяки широкому спектру інтерфейсів, включаючи RS-232, Ethernet та інші. Ці сканери також підтримують різноманітні типи штрихкодів, такі як 1D та 2D, що робить їх універсальними для різних застосувань.

Одним з аналогів SICK CLV62x є сканери штрихкода Zebra серії LS3408.

Вони також забезпечують високу точність сканування та можуть використовуватися в складних умовах. Проте, сканери Zebra зазвичай менш гнучкі в налаштуванні порівняно з SICK clv62x і можуть мати обмеженіший діапазон інтерфейсів.

Характеристики SICK CLV62x включають швидкість сканування до 1000 ліній за секунду, що забезпечує швидку обробку. Вони також мають компактний дизайн, що дозволяє легко вбудовувати їх у різні системи. Сканери можуть працювати в широкому діапазоні температур, що робить їх придатними для використання в умовах зовнішнього середовища.

Незважаючи на переваги, сканери SICK CLV62x мають деякі недоліки. Один з них – це вартість. Вони можуть бути дорожчими порівняно з деякими іншими моделями, що може бути стримуючим фактором для малих підприємств або стартапів. Іншим недоліком є обмежена підтримка для бездротових інтерфейсів, що може створювати проблеми у деяких умовах.

Принцип дії сканерів SICK CLV62x базується на використанні лазерної технології для зчитування штрихкодів. Лазерний промінь сканує поверхню об'єкта, і дані штрихкода обробляються внутрішніми алгоритмами сканера, перетворюючись на електронні сигнали. Ці сигнали потім передаються до системи управління через інтерфейси, такі як RS-232 або Ethernet.

Для під'єднання SICK CLV62x необхідно підключити сканер до джерела живлення та вибрати інтерфейс для передачі даних. Наприклад, використання RS-232 передбачає підключення кабелю до відповідного порту на сканері та системі управління. Якщо використовується Ethernet, сканер підключається до мережевого комутатора або маршрутизатора. Після під'єднання сканер необхідно налаштувати відповідно до вимог системи управління, що може включати конфігурацію параметрів сканування та інтерфейсів передачі даних (рис 2.8).

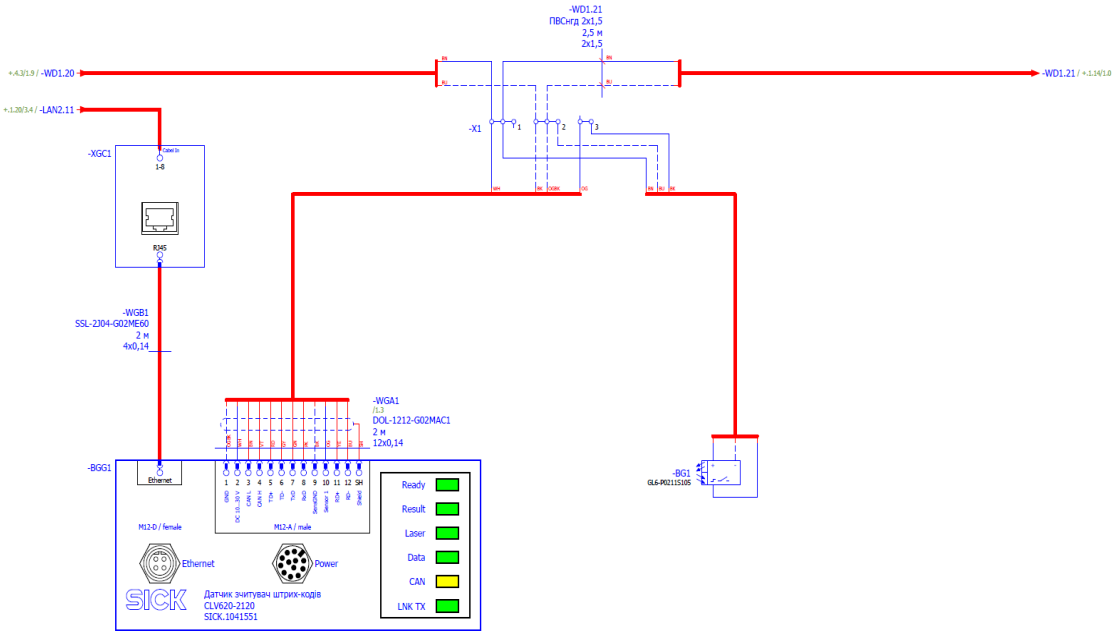


Рисунок 2.8 – Схема підключення сканера штрих-кодів SICK CLV62x

2.5 Логічна схема підключення конвеєрної системи до PLC

Після успішного підбору всього необхідного обладнання йде не менш важливий етап є підключення всіх цих елементів до контролерів, щоб в подальшому всім керувати. Для підключення буде використовуватися вита пара (Ethernet) буде під’єднуватися, як показано на рисунку 2.9. Для того щоб все коректна працювало потрібно під’єднати все в певній послідовності, середовище TIA Portal дозволяє в автоматичному режимі отримувати IP адреси кожного convaylinx AI2 [6].

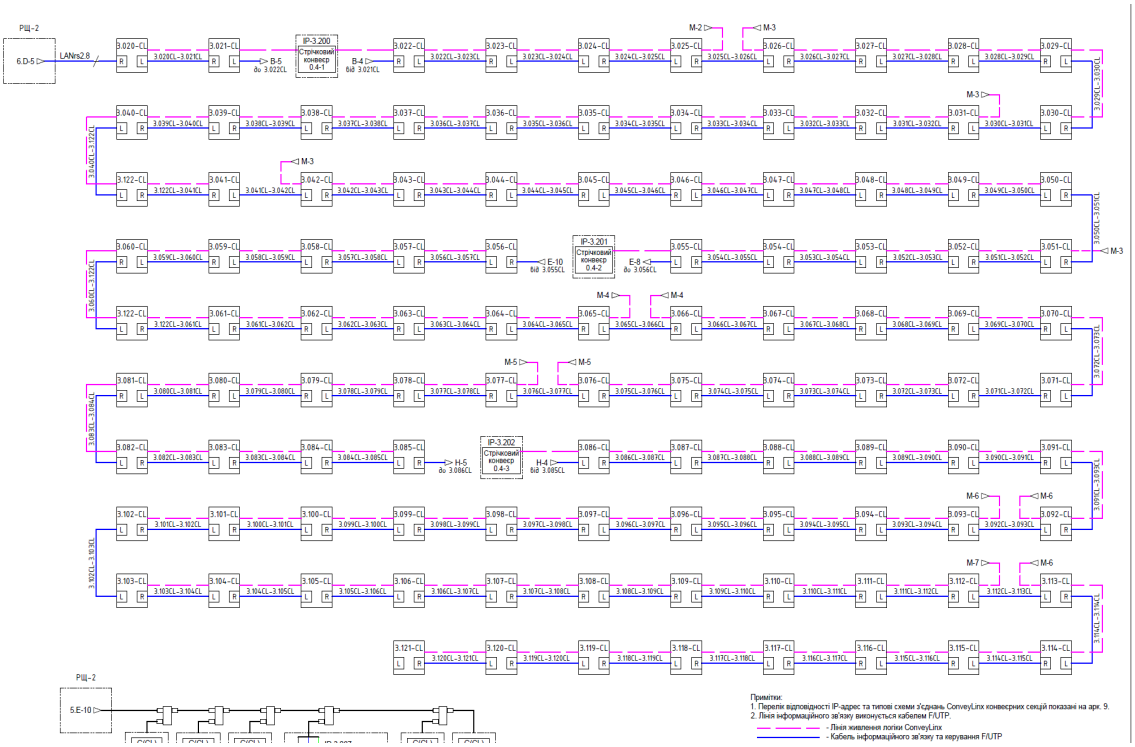


Рисунок 2.9 – Схема під’єднання convaylinx AI2 до PLC Siemens S1200

Після успішного підключення з’являється можливість підключитися до PLC та створити топологію підключення, та побачити послідовність підключення в програмі Siemens tiaportal v19. Щоб створити топологію в TIA Portal, необхідно додати всі пристрої, які ви плануєте використовувати, і визначити з'єднання між ними. Починаючи з нового або існуючого проекту, виберіть розділ "Пристрої та мережі" та перейдіть до "Виду мережі". У цьому режимі додайте пристрої, з якими будете працювати. Для з'єднання пристроїв використовуйте інструмент "з'єднання", вибираючи один пристрій, потім інший. Таким чином, з'єднайте всі пристрої, які повинні бути в мережі.

Для підтвердження правильності топології переконайтеся, що всі необхідні з'єднання присутні і відповідають фізичній структурі мережі. Після цього збережіть проект, щоб зберегти налаштування топології. Це основні кроки для створення топології в TIA Portal показано на рисунку 2.10.

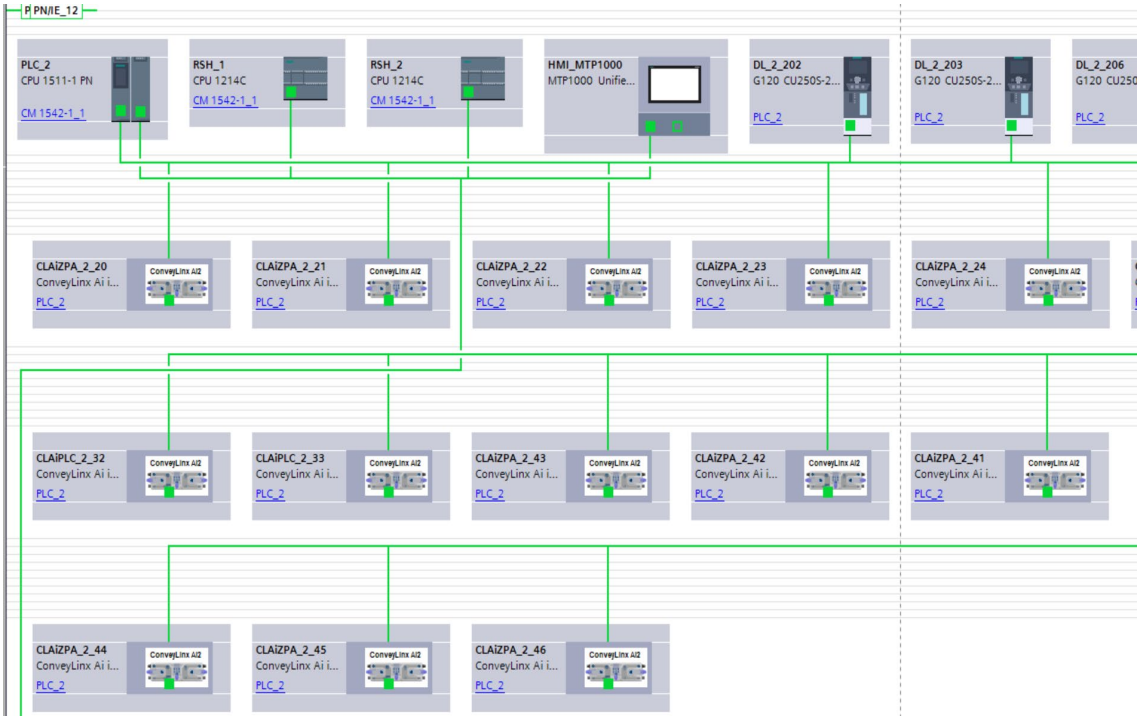


Рисунок 2.10 – Прописана топологія в середовищі tiaportal

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЧНОЇ КОНВЕЄРНОЇ СИСТЕМИ НА ОСНОВІ МОВИ РЕЛЄЙНОЇ ЛОГІКИ LAD

3.1 Підключення PLC Siemens S7-1214DC

Першочергово для керування всім етапом виробництва потрібно запрограмувати контролер Siemens S7-1214DC. Щоб коректно його налаштувати, його потрібно підключити до контуру безпеки, та інших елементів щиту для керування як показано на рисунку 3.1.

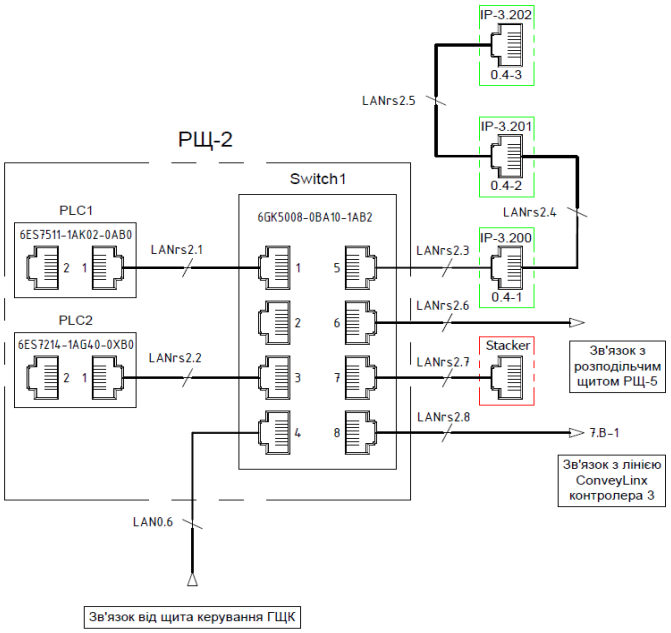


Рисунок 3.1 – Схема підключення контролеру Siemens S7-1214DC до інших частин виробництва

3.2 Підключення пінів PLC Siemens S7-1214DC

Після розуміння підключення контролеру до інших елементів виробництва, з’являється можливість програмування контролеру, для початку потрібно

ініціалізувати піни в середовищі TIA Portal. Щоб ініціалізувати піни (виходи або входи) на контролері Siemens у середовищі програмування TIA Portal, необхідно виконати наступні кроки:

1. Визначити імена пінів:

– зайти у Hardware Configuration (Конфігурація обладнання);

– знайти контролер та відкрийте Hardware Configuration;

– перегляньте список доступних модулів та визначте, які піни потрібно ініціалізувати [5];

2. Створити мітки (Tags):

– відкрити таблицю глобальних тегів (Global Tags Table) або локальних тегів (Local Tags Table) у програмі;

– створити нові теги для необхідних пінів. Вказати їм імена, тип даних та прив'язати до відповідних адрес. Наприклад, якщо потрібен вихід Q0.0, призначте тег з ім’ям "Output1" та адресою Q0.0.

3. Ініціалізація пінів у коді:

– повернутися до блоку програмування;

– вакористати оператор присвоєння або інші програмні структури для ініціалізації виходів або входів (рис 3.2).

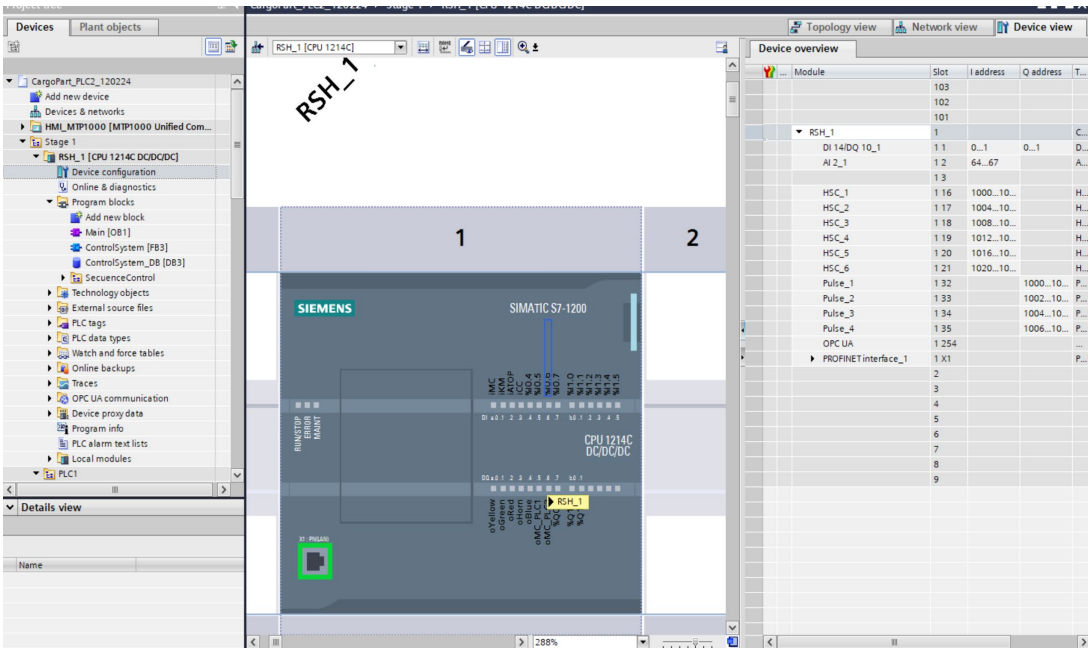


Рисунок 3.2 – Ініціалізація пінів контролера Siemens S7-1214DC

3.3 Створення організаційного блоку MAIN PLC Siemens S7-1214DC

У TIA Portal, сучасному середовищі розробки для автоматизації від Siemens, блоки OB (Організаційні блоки) є ключовими елементами програмування і управління логікою систем автоматизації. Вони функціонують як точка входу для програмного забезпечення контролера і визначають структуру та організацію програми. Існує кілька видів організаційних блоків, які виконують різні ролі в системі. Найпоширенішим OB є OB1, який є основним циклічним блоком, у якому виконується основна програма контролера.

Принцип роботи OB1 полягає у тому, що він працює в циклічному режимі, тобто повторюється з певним інтервалом часу, і кожен цикл виконується послідовно. Це забезпечує предсказувану і стабільну поведінку програмної логіки. Інші OB можуть бути пов'язані з різними подіями, такими як перезавантаження системи, помилки, таймери, або переривання.

Програмування на мові LAD (Ladder Diagram) – це графічний спосіб представлення логічних структур програми, який нагадує електричну схему з релейними контактами. В LAD кожен елемент програми має візуальне уявлення, де логічні операції представлені у вигляді "сходинок", з лініями живлення, контактами, котушками та іншими елементами. Ці елементи використовуються для реалізації логіки, яка є подібною до логіки релейних схем.

У LAD можна використовувати контакти для моделювання логічних входів, які можуть бути нормально відкритими (NO) або нормально закритими (NC), а котушки представляють виходи. Логічні операції, такі як І (AND), АБО (OR), НЕ (NOT), реалізуються шляхом комбінування контактів у відповідних конфігураціях.

Коли ви працюєте з OB1 у LAD, ви розміщуєте контакти, котушки та інші логічні блоки у потрібному порядку для досягнення необхідної логічної послідовності. Наприклад, для створення логіки "І", ви розміщуєте контакти послідовно, а для "АБО" – паралельно. Котушки вказують, який вихід буде

активований у разі виконання певних умов.

Відповідно, програма в OB1 на мові LAD має циклічний характер: кожен цикл програми виконується зверху вниз, перевіряючи умови і встановлюючи виходи. Інші OB можуть бути запрограмовані для різних подій, як-от запуск по таймеру або обробка аварійних ситуацій. Ключовим для програміста є розуміння структури системи, правильного застосування логічних блоків та забезпечення стабільного циклічного виконання програми. На рисунку 3.3 виконано ініціалізацію пінів, проходить циклічна перевірка на включення пінів контролера 0.0, 0.2, 0.3.

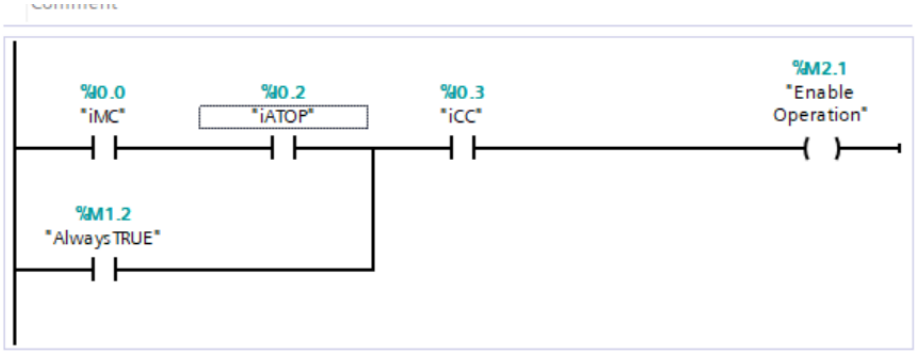


Рисунок 3.3 – організаційний блок Main Network1 Siemens S7-1214DC

Якщо живлення присутнє на всіх цих пінах, тільки тоді наш контролер зможе продовжити своє включення та запустити наступний організаційний блок, який зображений на рисунку 3.4.

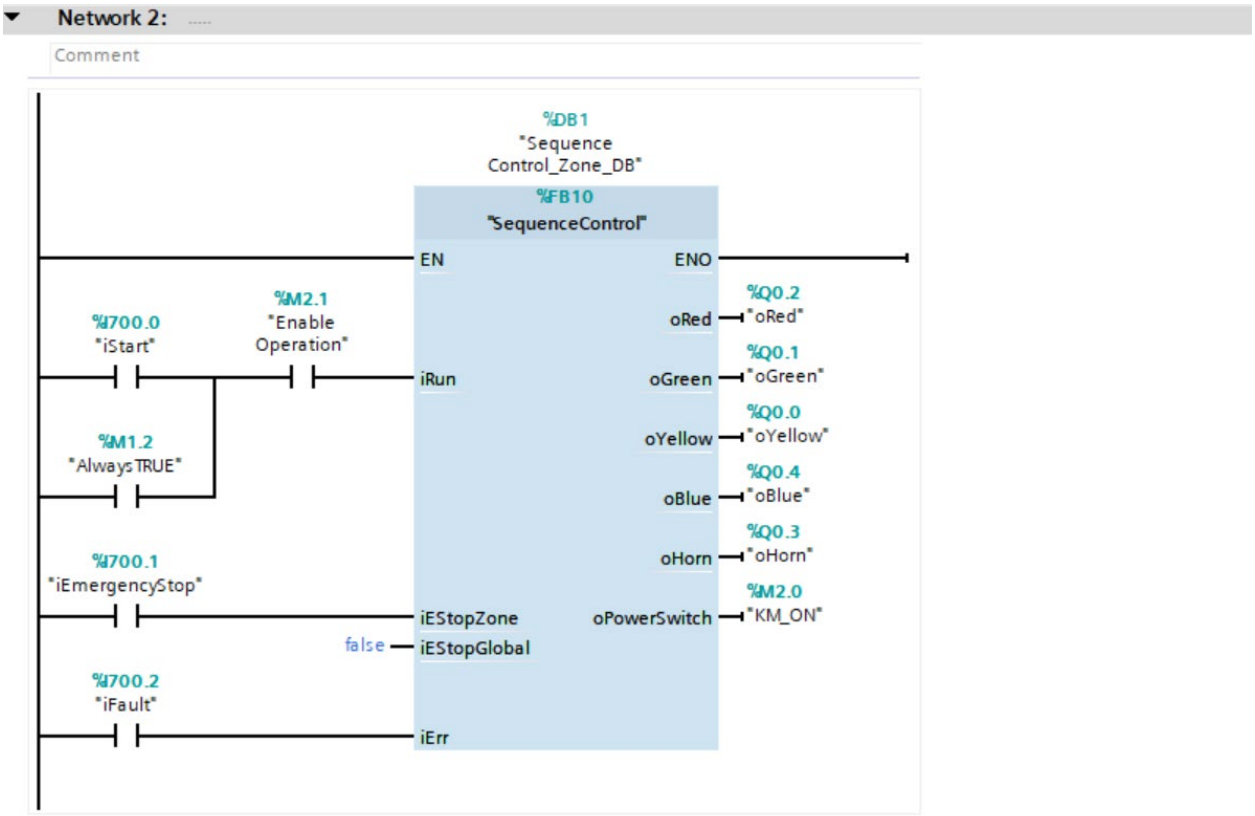


Рисунок 3.4 – Організаційний блок Main Network2 Siemens S7-1214DC

На рисунку 3.4 можна побачити основний, навіть можна сказати головний з організаційних блоків він відповідає за перевірку безпеки, вімкнення саме цього контуру безпеки, перевірку світло-звукової сигналізації, виведення кольору та включення контуру безпеи щита керування, на рисунку 3.5 зображено ініціалізація пінів контуру безпеки.

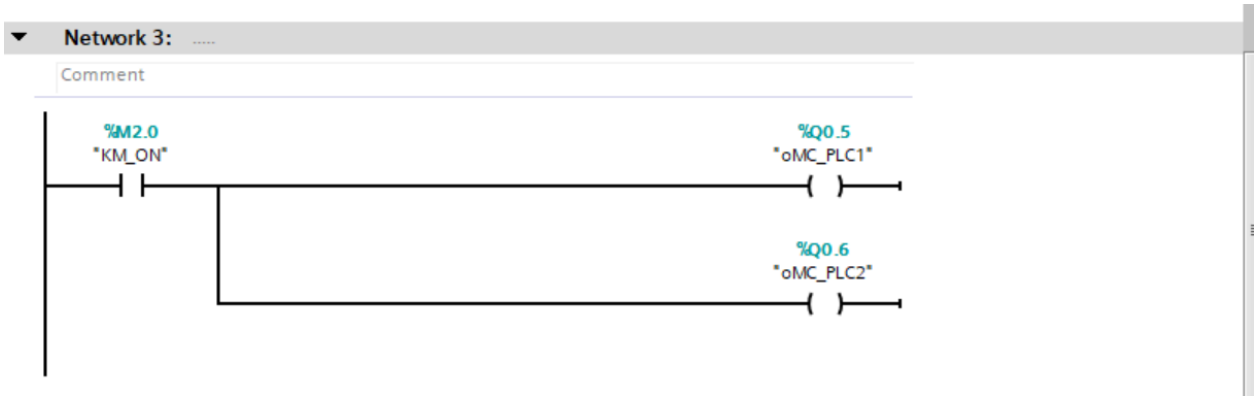


Рисунок 3.5 – Організаційний блок Main Network2 Siemens S7-1214DC

На рисунку 3.5 була проведена ініціалізація пінів контуру безпеки, це дозволить подати живлення на PLC Siemens S7-1500.

3.4 Створення функціональних блоків PLC Siemens S7-1214DC

Функціональний блок (Function Block, FB) у програмуванні контролерів (PLC) Siemens — це тип блоку програмування, який дозволяє створити багаторазово використовуваний модуль коду з параметрами вводу/виводу. Він зберігає внутрішній стан через використання екземплярів даних (Data Blocks, DB), що робить його відмінним від звичайних організаційних блоків (Organization Blocks, OB) або блоків функцій (Function Blocks, FC), які не зберігають стан.

Функціональні блоки дозволяють створювати модульні, багаторазово використовувані та інкапсульовані частини програми, які можуть взаємодіяти між собою через параметри та глобальні дані. порядок створення функціональних блоків:

1. Додавання функціонального блоку:

– в дереві проєкту (Project Tree) знайти секцію "Program Blocks";

– натиснути правою кнопкою миші та обрати "Add new block" (Додати новий блок);

– обрати "Function Block" (Функціональний блок);

– ввести ім'я для нового функціонального блоку, наприклад, "myfunctionblock";

– обрати мову програмування (LAD, FBD, STL, SCL), в заленості від цілей проєкту.

2. Створення екземпляру даних (Data Block): коли створили функціональний блок, TIA Portal автоматично створює екземпляр даних, пов'язаний з цим блоком. Цей екземпляр даних зберігає стан змінних та дані для конкретного виклику функціонального блоку.

3. Програмування функціонального блоку:

– відкриття функціональний блок, який був щойно створений;

– додаємо необхідні параметри вводу/виводу (In/Out parameters), використовуючи секцію "Interfaces" (Інтерфейси) або таблицю "Parameter";

– програмування функціональний блок, як будь-який інший блок, використовуючи відповідні логічні елементи, операції, та інші компоненти коду.

4. Виклик Функціонального Блоку:

– перейти до організаційного блоку (OB), наприклад, OB1;

– додати новий виклик функціонального блоку (FB Call);

– обрати ім'я вашого функціонального блоку;

– обрати або створити екземпляр даних, якщо його ще немає;

– встановити параметри вводу/виводу для функціонального блоку.

Функціональні блоки є важливими для побудови складних систем, де потрібно мати гнучкі, повторно використовувані компоненти коду з можливістю зберігання внутрішнього стану.

Проаналізувавши вимоги проекту було створено наступні функціональні блоки для PLC Siemens S7-1214DC.

Функціональний блок controlsystem (FB3):

```
#TON_Load(IN:="alwaystrue" AND NOT "firstscan",
      PT:=t#30s,
      Q=>#loadprocess);
#delaystartsystem(IN:=#startsystem,
            PT:=T#20s);
```

Перша функція `TON_Load` використовує таймер із затримкою 30 секунд. Вона активується, коли умова `IN` є `TRUE`. У цьому випадку, умова визначається як поєднання значення `alwaystrue` та заперечення `firstscan`, тобто це не перше сканування програмного циклу. Як тільки таймер активується, він починає відлік часу, і коли досягає 30 секунд, встановлює вихід `Q` в значення `TRUE`. Вихід `Q` пов'язаний зі змінною `#loadprocess`, що означає, що після завершення першого сканування і проходження 30 секунд, змінна `#loadprocess`

буде активована. Цей таймер може бути використаний для ініціалізації певних процесів із затримкою після початкового запуску системи.

Друга функція `delaystartsystem` також використовує таймер, але з затримкою 20 секунд. Умова активації цього таймеру визначається значенням `#startsystem`. Коли `#startsystem` стає `TRUE`, таймер починає відлік. Після 20 секунд вихід таймеру (`Q`) стає `TRUE`, що вказує на активацію певної дії або процесу. Цей таймер може бути використаний для відкладеного запуску системи або певної операції, щоб забезпечити час на інші ініціалізації чи завдання перед початком основної дії.

Загалом, обидва таймери створюють затримки перед виконанням дій, але з різним часом. Це дозволяє виконувати дії в контрольованому порядку, уникнути конфліктів або забезпечити поступовий запуск системи.

Функціональний блок sequencecontrol(FB10):

```
//Таймер завантаження системи
#loaddelay(IN:="alwaystrue",
        PT:=t#30s,
        Q=>#sloadprocess);
//Таймер запуску системи
#startdelay(IN := #sstartsystem (* #irun AND NOT #iestopglobal AND NOT #iestopzone *),
         PT := #ctimestart);
//Таймер зупинки системи
#stopdelay(IN := NOT #irun AND #sstop,
         PT := #ctimestop);
//////////////////////////ЗАПУСК////////////////////////////////
////Вмикання передпускової сигналізації
IF #sloadprocess AND #irun AND NOT #iestopglobal AND NOT #iestopzone AND NOT #sworksystem THEN
   #sstartsystem := TRUE;
```

```
    #sstop := TRUE;
    #ored := FALSE;
    #oyellow := TRUE;
    #ogreen := FALSE;
    #ohorn := TRUE;
    #opowerswitch := FALSE;
END_IF;
//Вмикання живлення
IF #startdelay.Q THEN
    #sstartsystem := FALSE;
    #sworksystem := TRUE;
    #sstop := FALSE;
    #ored := FALSE;
    #oyellow := FALSE;
    #ohorn := FALSE;
    #opowerswitch := TRUE;
END_IF;
//////////////////////////////ПОМИЛКА//////////////////////////////
IF #ierr AND #sworksystem THEN
    #oblue := TRUE;
    #ogreen := FALSE;
ELSIF NOT #ierr AND #sworksystem THEN
    #oblue := FALSE;
    #ogreen := TRUE;
END_IF;
//////////////////////////////ЗУПИНКА//////////////////////////////
IF NOT #irun AND #sworksystem AND NOT #iestopzone AND NOT #sstartsystem
THEN
    #sstartsystem := FALSE;
    #sworksystem := FALSE;
```

```
    #sstop := TRUE;
    #ored := FALSE;
    #oyellow := FALSE;
    #ogreen := TRUE;
    #ohorn := TRUE;
    //#opowerswitch := TRUE;
END_IF;
IF #stopdelay.Q AND #sstop THEN
    #sstartsystem := FALSE;
    #sworksystem := FALSE;
    #sstop := FALSE;
    #ored := FALSE;
    #oyellow := FALSE;
    #ogreen := FALSE;
    #ohorn := FALSE;
    #opowerswitch := FALSE;
END_IF;
//////////////////////////АВАРІЙНИЙ РЕЖИМ//////////////////////
//Аварійна зупинка
IF #iestopzone OR #iestopglobal THEN
    #sstartsystem := TRUE;
    #sworksystem := FALSE;
    #sstop := FALSE;
    #ored := TRUE;
    #oyellow := FALSE;
    #ogreen := FALSE;
    #ohorn := TRUE;
    #opowerswitch := FALSE;
ELSE
    #ored := FALSE;
```

```
END_IF;
//Таймер аварійної зупинки
#estopdelay(IN := #iestopzone OR #iestopglobal,
          PT := #ctimealarm);
//Аварійна сигналізація
IF #estopdelay.Q THEN
   #ohorn := FALSE;
END_IF;
```

Цей код описує логіку керування системою, використовуючи різні таймери та умови для запуску, зупинки та аварійного режиму.

Спочатку, таймер завантаження системи (`#loaddelay`) починає 30-секундний відлік при активації з вічно істинною умовою. Це дозволяє системі виконати необхідні дії під час завантаження. Коли таймер спрацьовує, система може переходити до наступного кроку.

Для запуску системи використовується інший таймер (`#startdelay`), який активується за умови, що система готова до запуску (сигнал `#sstartsystem`), інші умови запуску виконані, і немає екстреної зупинки. Після того як таймер спрацьовує, система вмикає живлення, припиняє передпускову сигналізацію, і переходить в режим роботи.

При зупинці системи, якщо сигнал `#sstop` і умова відсутності виконання (`NOT #irun`), використовується таймер затримки зупинки (`#stopdelay`). Якщо умови виконані, система зупиняється, активуючи відповідні сигнали, як-от гудок та індикатори, показуючи зупинку. Після завершення таймера, система повністю відключається.

Аварійний режим активується, якщо виникає одна з умов екстреної зупинки: `#iestopzone` або `#iestopglobal`. У цьому випадку система відключається, червоний індикатор вмикається, гудок сигналізує, а система переходить у безпечний стан. Таймер аварійної зупинки (`#estopdelay`) використовується для зменшення інтенсивності сигналізації після певного часу.

Помилки також обробляються умовно: якщо є помилка під час роботи системи (`#ierr`), синій індикатор сигналізує про проблему. Якщо помилки немає, зелений індикатор вказує, що система працює нормально [7].

Таким чином, код створює механізм керування системою, використовуючи таймери та умови для забезпечення безпечного запуску, зупинки та аварійного режиму, включаючи відповідні візуальні та звукові індикатори для кожного стану.

3.5 Тестування працездатності та коректної роботи контролеру Siemens S7-1214DC

Правильне тестування з використанням PLC Simulator у Siemens TIA Portal є критично важливим етапом у розробці автоматизованих систем. Перш ніж розпочати, необхідно створити проект у TIA Portal та налаштувати його для конкретного PLC. Напишіть програму у мові програмування, яка підтримується TIA Portal, таку як Ladder Logic або Structured Text, забезпечивши її відповідністю вимогам проекту.

Після цього PLC Simulator у проекті та запустіть симуляцію. Це дозволить відтворити роботу програми у віртуальному середовищі. Підключіться до PLC Simulator через TIA Portal, щоб моніторити роботу програми та взаємодіяти з нею у реальному часі.

Ретельне тестування програми, всі її функції та взаємодію з введеними даними. Виявлені помилки виправляються та повторюється тестування. Після успішного завершення тестування здійснюється валідацію та оптимізацію програми, враховуючи будь-які виявлені під час тестування недоліки чи можливі покращення. Такий підхід допоможе забезпечити надійність та ефективність вашої автоматизованої системи перед її впровадженням у реальне середовище.

Для цього було на рисунках 3.6-3.10 продемонстровано ввесь цикл тестування з його подальшим обґрунтуванням.

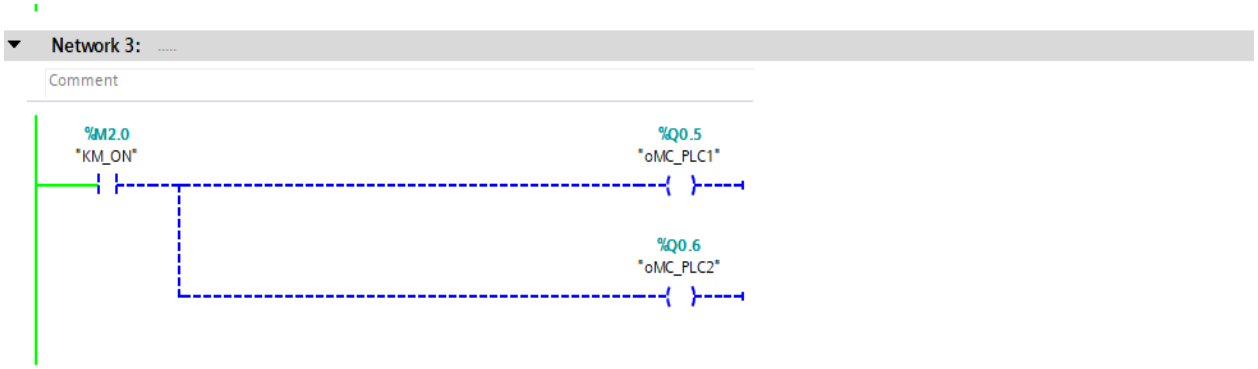


Рисунок 3.6 – Вимкнений контур безпеки контролеру S7-1214DC

Щоб увімкнути контур безпеки потрібно подати живлення на пін з адрессою %M2.0. Для того щоб подати на нього живлення ми маємо подати живлення на пакі адрес як %I0.0:P, %I0.1:P, %I0.2:P, %I0.3:P, %I7000:P

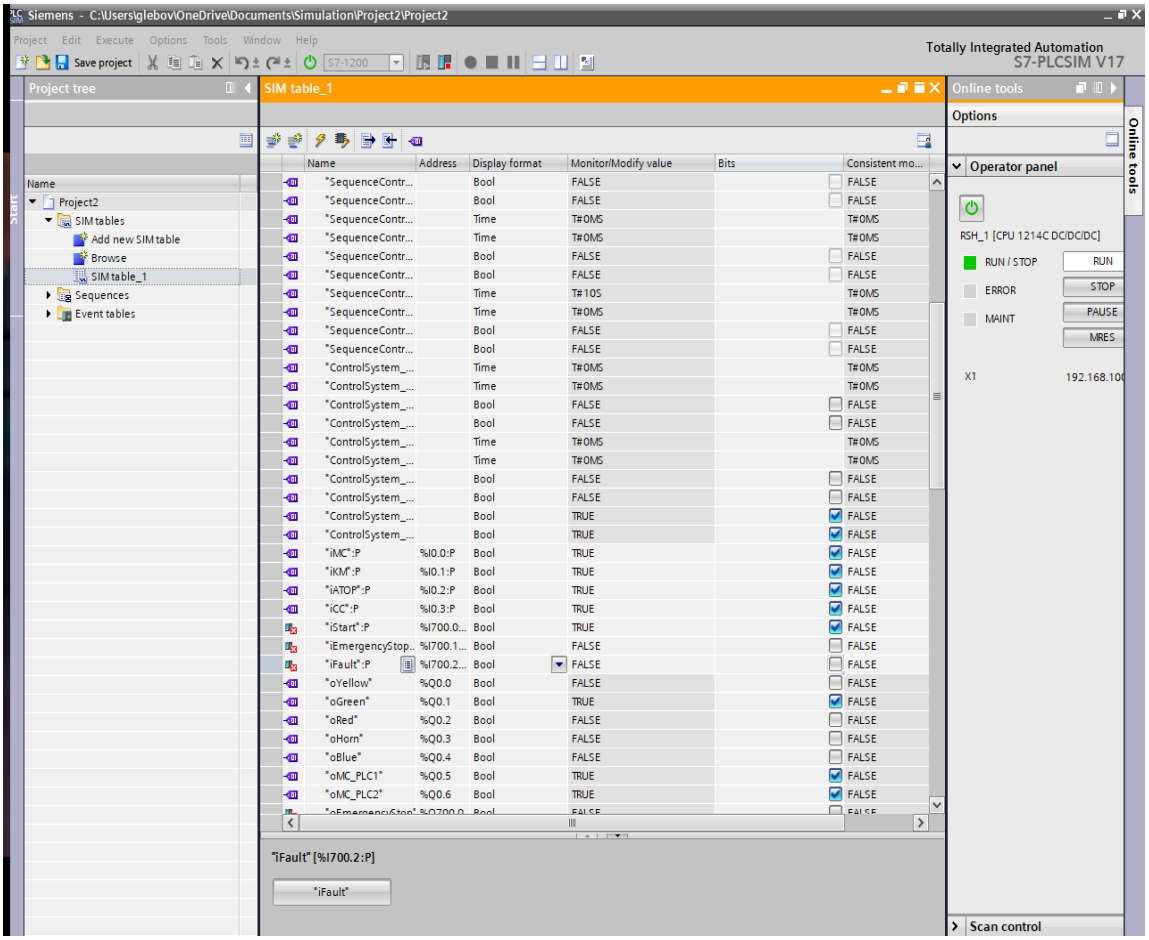


Рисунок 3.7 – Подача сигналу на адрес %I0.0:P, %I0.1:P, %I0.2:P, %I0.3:P, %I7000:P контролеру S7-1214DC

Після успішної подачі живлення на ці піни ми можемо побачити на схемах релейної логіки, як живлення успішно пішло на контур безпеки, ввімкнулась світлозвукова сигналізація зеленого кольору що символізує відсутність помилок.

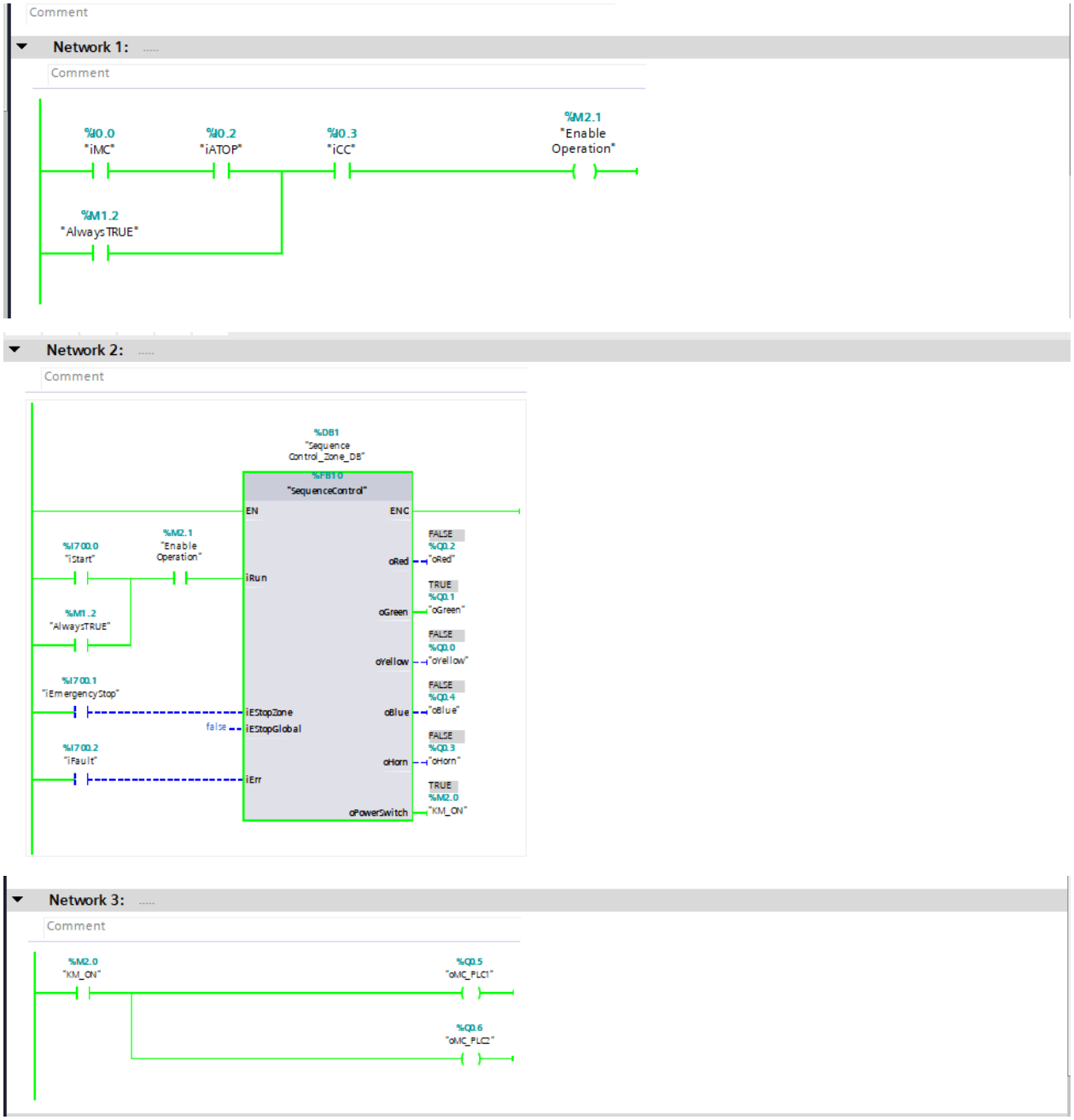
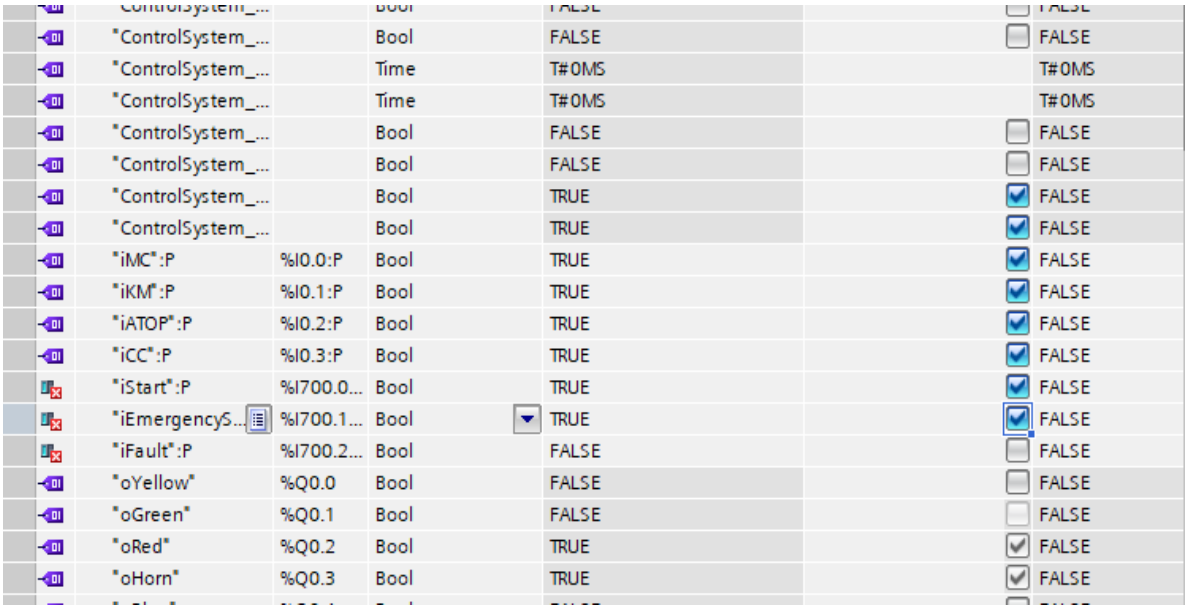


Рисунок 3.8 – Результат увімкнення адрес %I0.0:P, %I0.1:P, %I0.2:P, %I0.3:P, %I7000:P контролеру S7-1214DC

На рисунку ми можемо побачити всі перевірки, та відсутність помилок, це є першим та ключовим етапом в подальшому налаштуванні контролерів S7-1511-1

PN PLC1 – PLC2. Також завдяки симуляції є можливість протестувати аварійні режими, та режими помилок щоб перевірити реакцію системи на такі події. Для того щоб це зробити в PLC SIM, ми має надати статус True значенню "iemergencystop":P за адресою %I700.1:P.



	"ControlSystem_...		Bool	FALSE	FALSE
	"ControlSystem_...		Time	T#0MS	T#0MS
	"ControlSystem_...		Time	T#0MS	T#0MS
	"ControlSystem_...		Bool	FALSE	FALSE
	"ControlSystem_...		Bool	FALSE	FALSE
	"ControlSystem_...		Bool	TRUE	FALSE
	"ControlSystem_...		Bool	TRUE	FALSE
	"iMC":P	%I0.0:P	Bool	TRUE	FALSE
	"iKM":P	%I0.1:P	Bool	TRUE	FALSE
	"iATOP":P	%I0.2:P	Bool	TRUE	FALSE
	"iCC":P	%I0.3:P	Bool	TRUE	FALSE
	"iStart":P	%I700.0...	Bool	TRUE	FALSE
	"iEmergencyS...	%I700.1...	Bool	TRUE	FALSE
	"iFault":P	%I700.2...	Bool	FALSE	FALSE
	"oYellow"	%Q0.0	Bool	FALSE	FALSE
	"oGreen"	%Q0.1	Bool	FALSE	FALSE
	"oRed"	%Q0.2	Bool	TRUE	FALSE
	"oHorn"	%Q0.3	Bool	TRUE	FALSE

Рисунок 3.9 – Подача сигналу на адрес %I700.1:P контролеру S7-1214DC

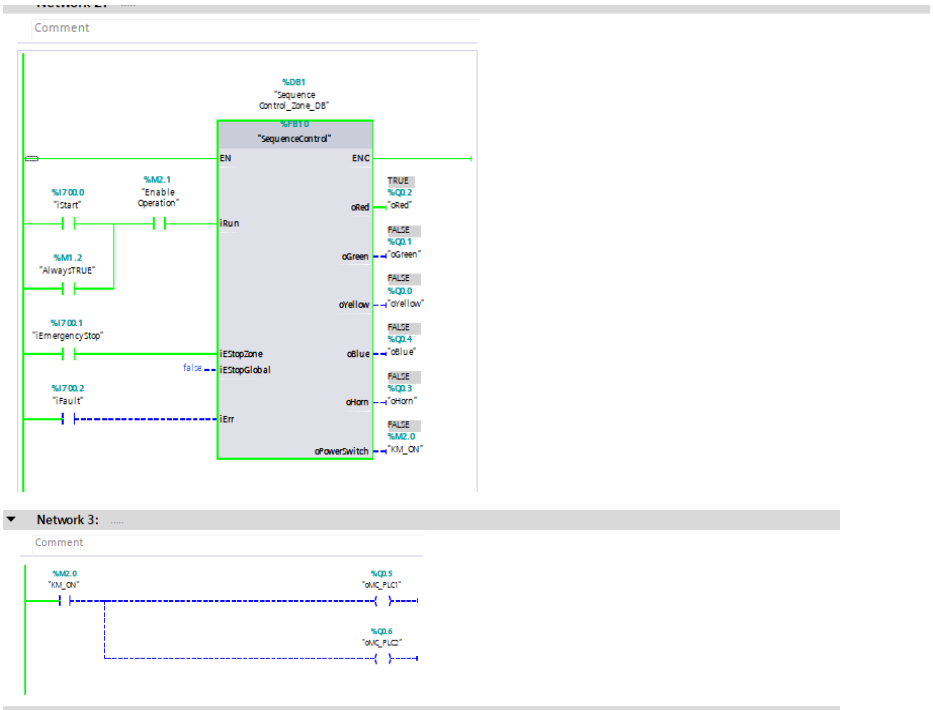


Рисунок 3.10 – Результат подачі сигналу на адрес %I700.1:P контролеру S7-1214DC

Дивлячись на результат можна побачити, що контур безпеки був вимкнений, та сигналізація змінила колір на червоний, і увімкнулася звукова сигналізація. Також є можливість перевірити режим несправності, в цьому режимі система буда продовжувати свою роботу, але індикація змінить колір на синій що буде символізувати, що програма працює з помилками які не критично вплинули на роботу системи, але на них потрібно звернути увагу на рис. 3.11 – 3.12. Такими помилка може бути, знаходження товару з некоректним штрих кодом, і з неможливістю зчитати його система відправіть його в штрафну зону і увімкне саме даний режим роботи. За це відповідає "ifault":P, за адресою %I700.2:P [8].

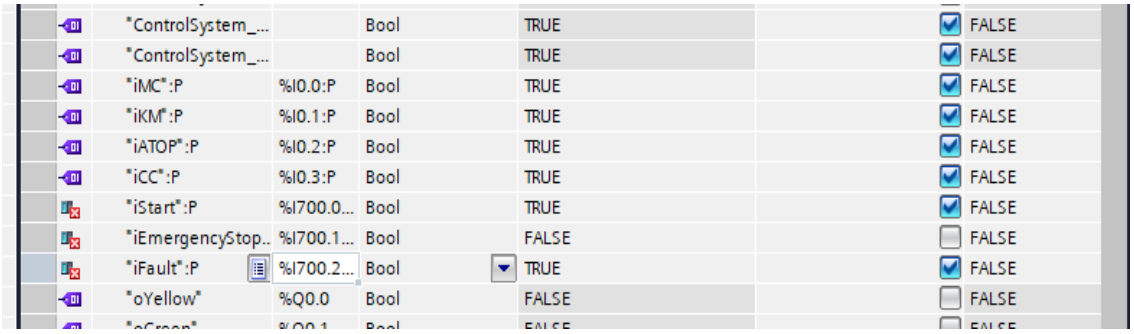


Рисунок 3.11 – Подача сигналу на адрес %I700.1:P контролеру S7-1214DC

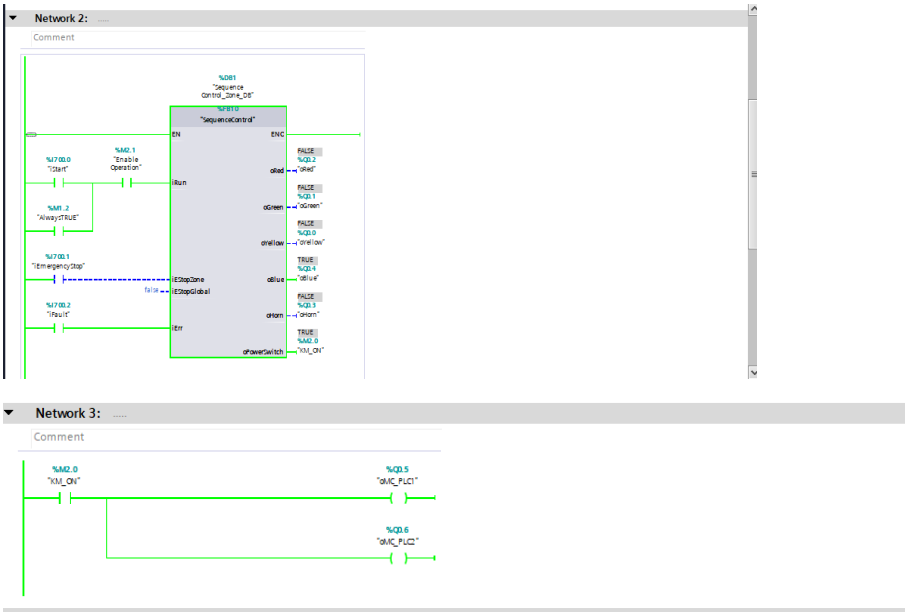


Рисунок 3.12 – Результат подачі сигналу на адрес %I700.1:P контролеру S7-1214DC

Ці ключові перевірки дозволили зробити систему надійною, розсортувати помилки та аварії по рівням їх критично. Все це підготувало основу для подальшого налаштування контролерів Siemens S7-1511-1 PN PLC1 – PLC2, вони будуть ключовими в роботі системи та відповідати за роботу всіх конвелінсів, та двигунів по системі.

3.6 Створення організаційного блоку MAIN PLC Siemens S7-1511-1 PN PLC1 – PLC2

Як зазначено в створенні організаційних блоків PLC Siemens S7-1214DC, це є важливим елементом створення програм для контролерів. Для контролерів Siemens S7-1511-1 PN PLC1 були створені наступні організаційні блоки:

– ТПР 112, RAT30 1.20, ТПР 113, RAT30 1.28 – вони відповідають роботу та свівілів. Основні потреби до цього блоку є перевірки помилок, перевірки акселерації, положення товарів на свівілах та інші (рис 3.13).

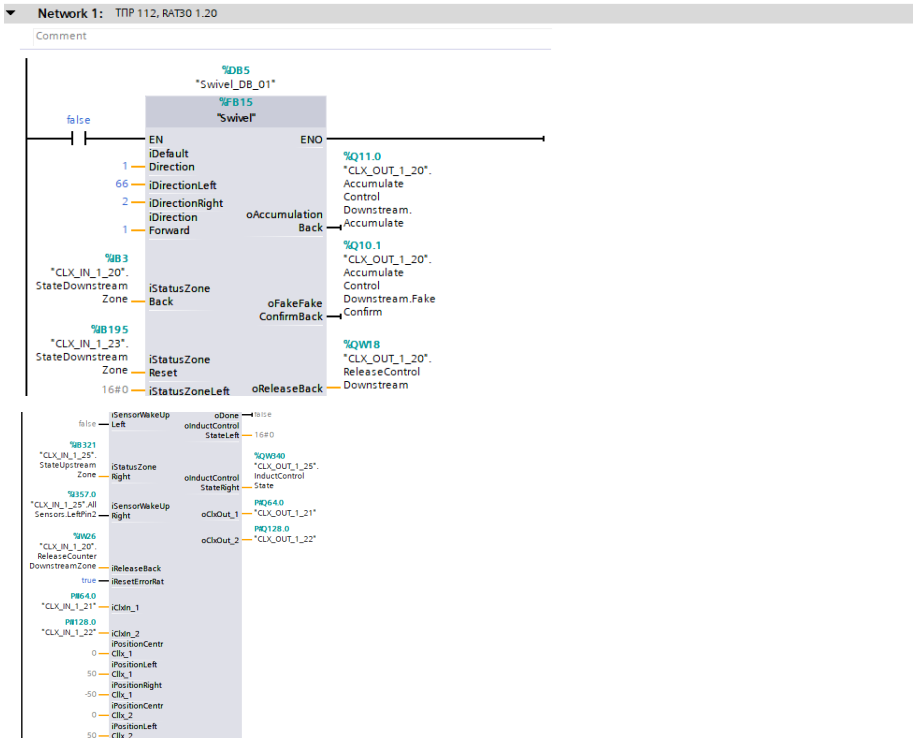


Рисунок 3.13 – Організаційні блоки ТПР 112, RAT30 1.20 контролеру S7-1511-1 PN PLC1.

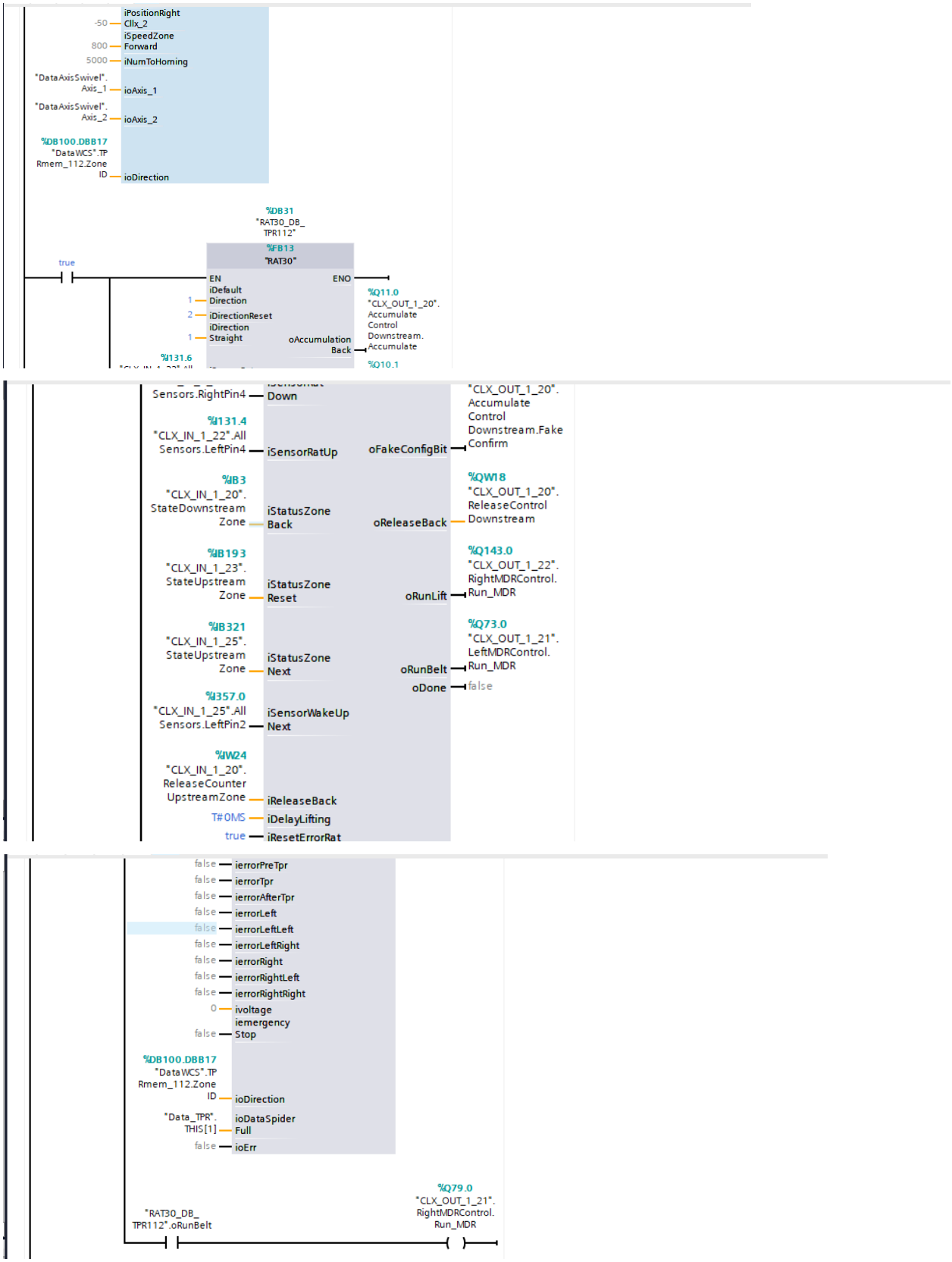


Рисунок 3.13, аркуш 2

Таких організаційних блоків потрібно 2 штуки в залежності від IP-адреси свівілу. Далі потрібно створити організаційні блоки для керування двигунами стрічкових конвеєрів. Нам потрібно 9 таких блоків для двох PLC1 та 4 штуки для PLC2. Вони мають виконувати переміщення товару з другого рівня на нижній та внутрішні перевірки на помилки, це можно побачити на рисунку 3.14.

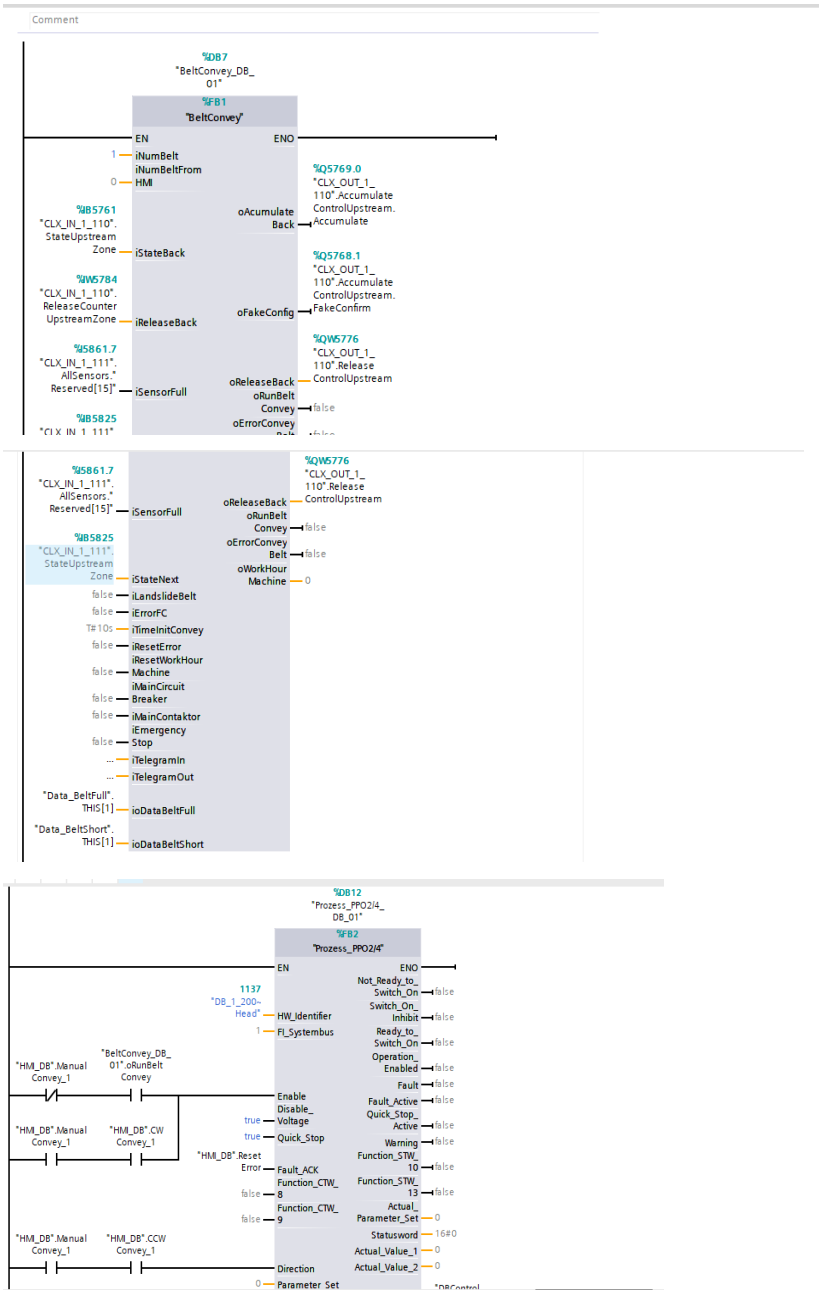


Рисунок 3.14 – Організаційні блоки керування двигунами контролеру S7-1511-1 PN PLC1-PLC2.

Після успішного додавання цих блоків, потрібно додати блоки вливання з однієї лінії конвеєра в іншу. Таких вливань має бути 7 штук. Керуватися вони будуть таким чином: 5 вливань будуть контролюватися PLC2, а 2 вливання - PLC1. Така відмінність в кількості блоків є дуже важливою через технічні особливості контролерів S7-1511-1 PN. Рекомендована кількість пристроїв для контролю, зазначена в документації, складає до 120 пристроїв. Всі інші пристрої можуть працювати некоректно або з досить великими затримками, що є критичним для виробництва на рис. 3.15– 3.16.

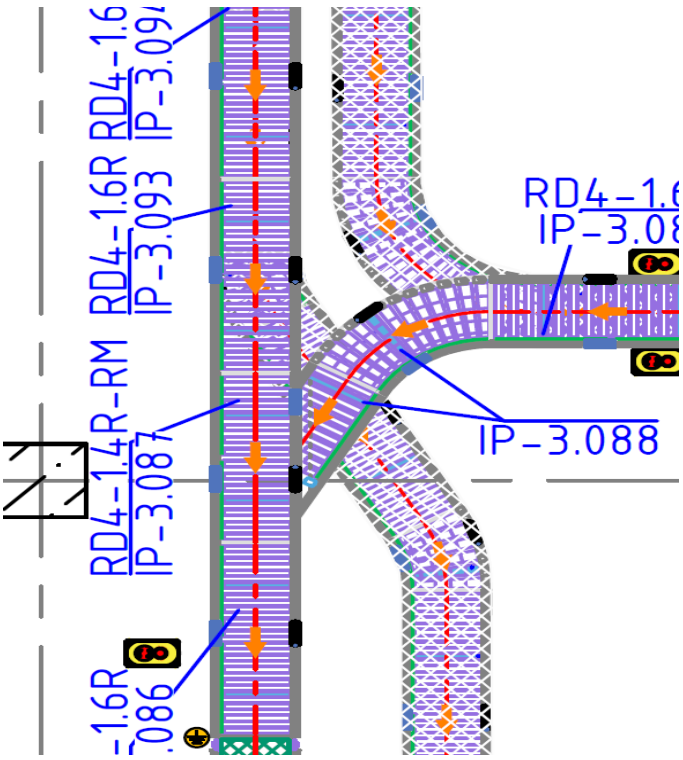


Рисунок 3.15 – Вливання в систему на схемі

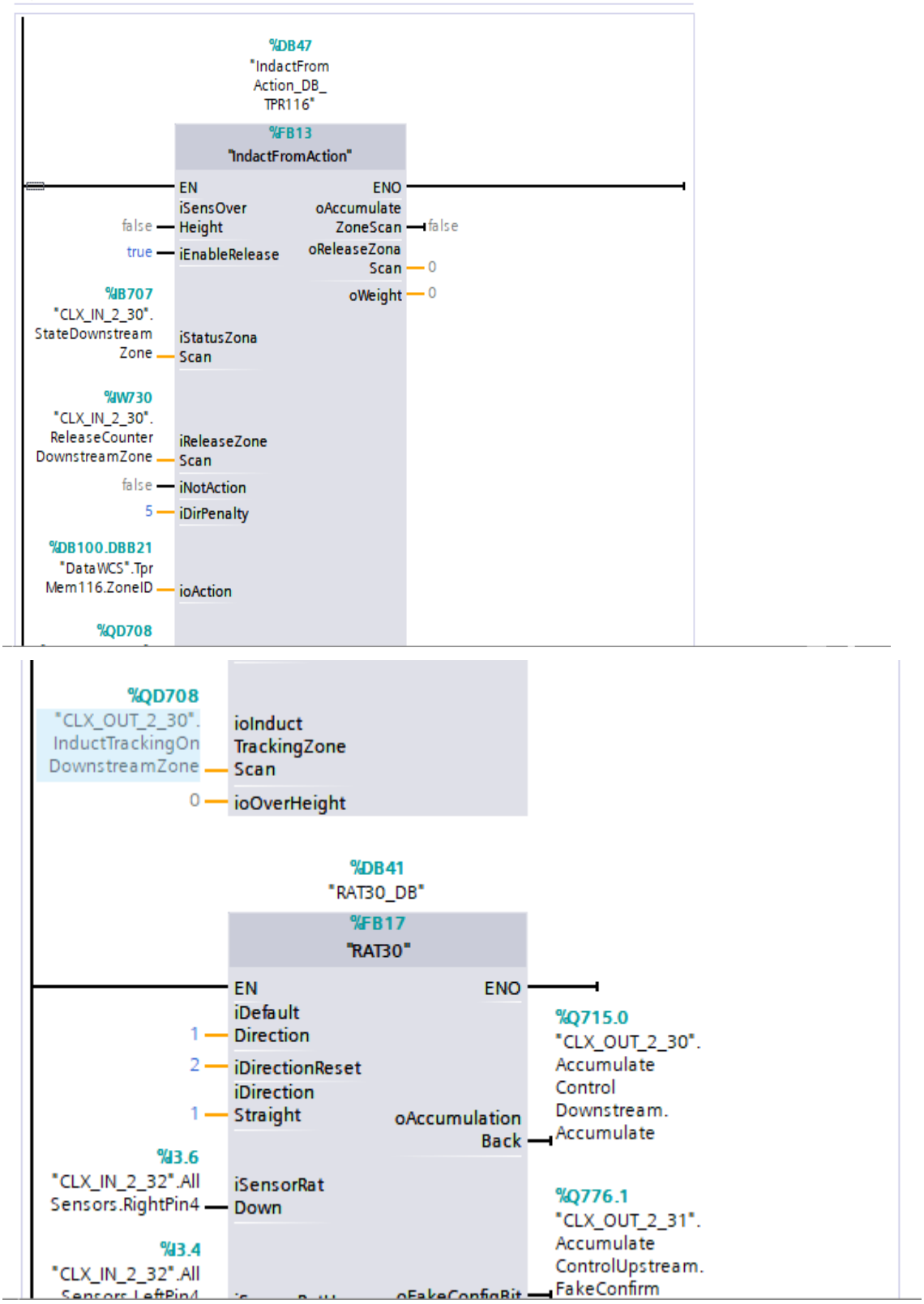


Рисунок 3.16 – Організаційні блоки вливаннями TPR 114, 115, 116, 118, 121, 122. Контролеру S7-1511-1 PN PLC1-PLC2.

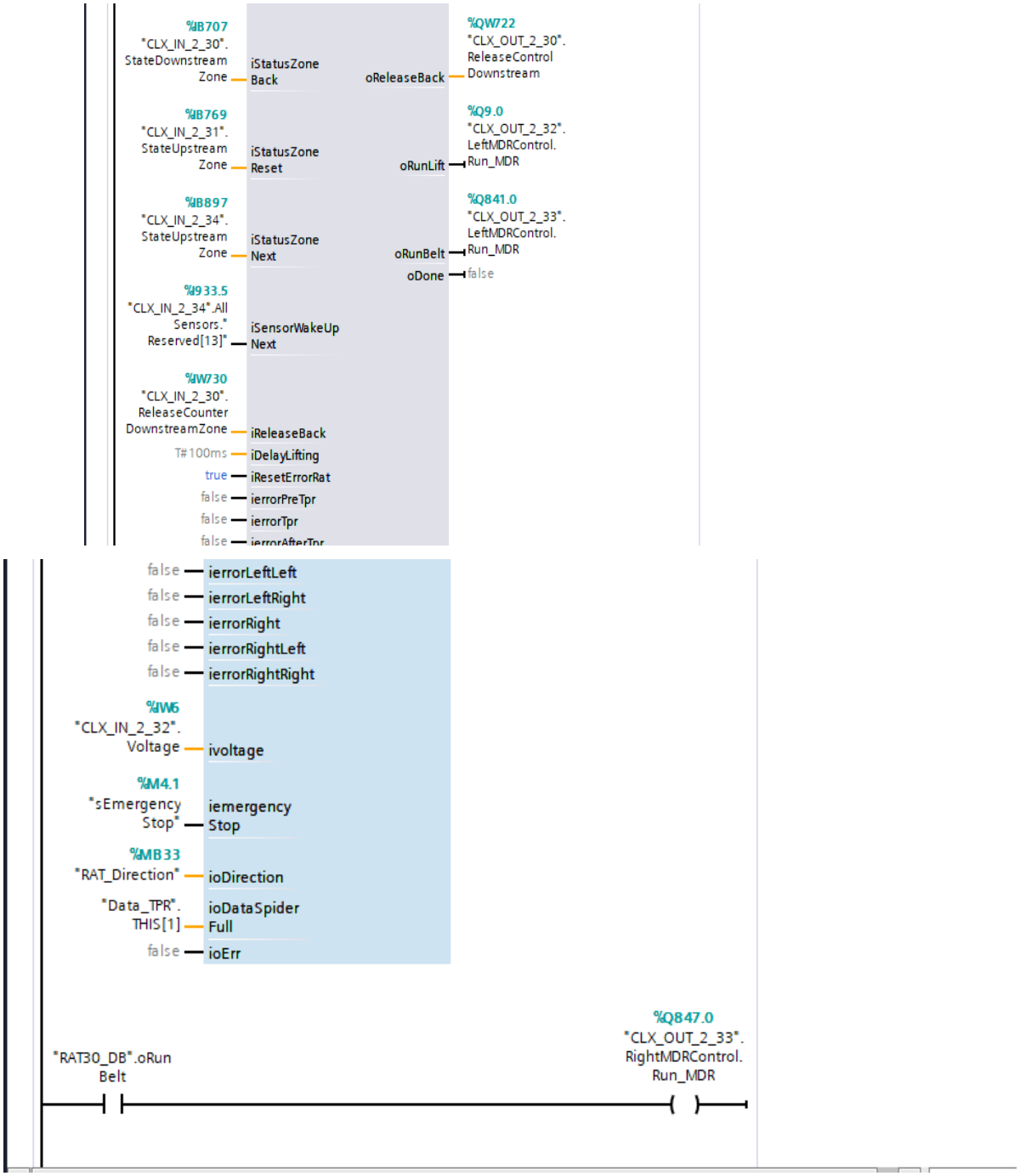


Рисунок 3.16, аркуш 2

Наступник кроком буде додавання блоків керування ліфтами, це довзволить переміщувати товар по зоні сортування, та відправляти його діл. Таких зон буде 4 виходячи з цього нам потробно 8 ліфтів, які будуть керуватися по 4 штуки на PLC. На рисунку 3.17 приклад зони сортування та на рисунку 3.18 організаційний блок керування ліфтом.

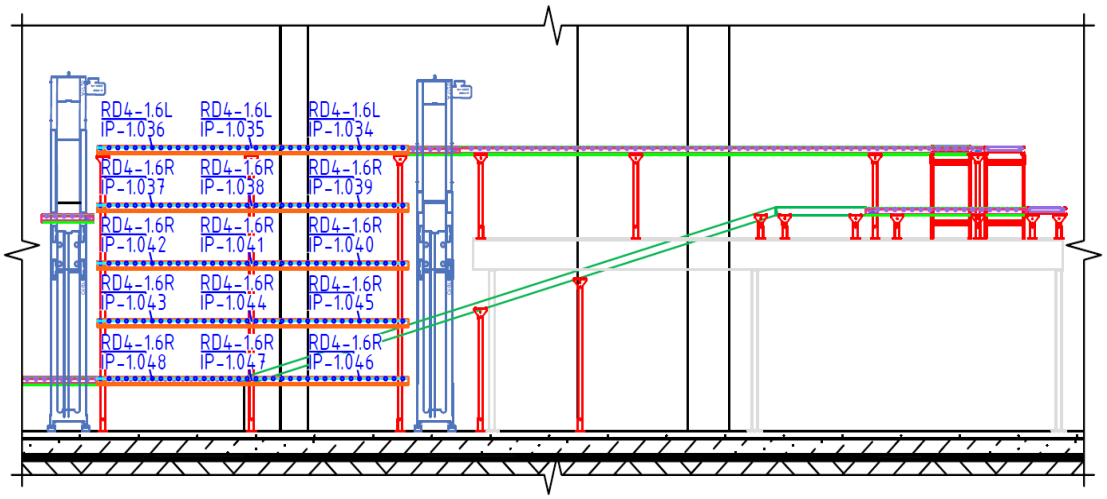


Рисунок 3.17 – Схема зони сортування.

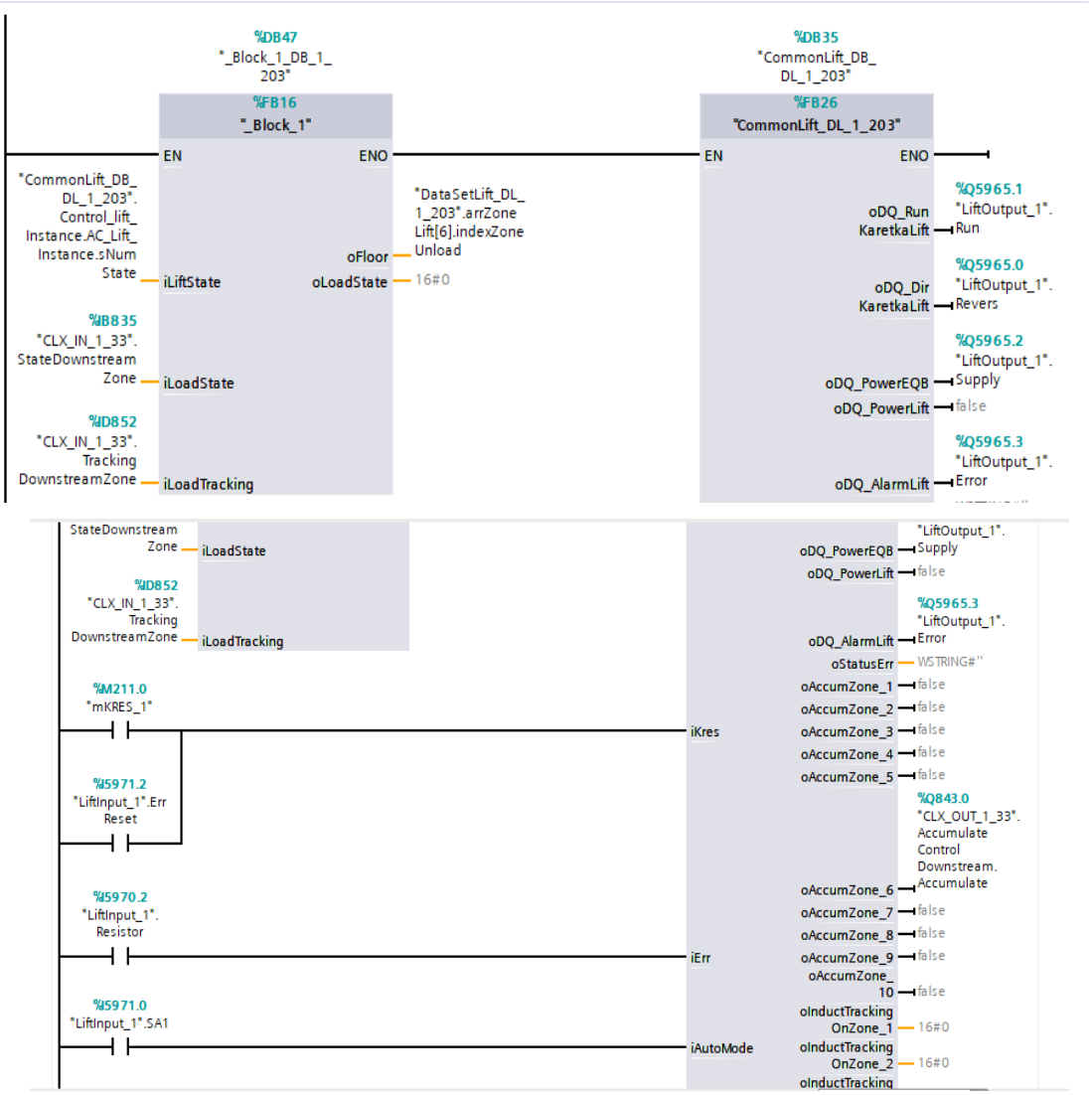


Рисунок 3.18 – Організаційні блоки керування ліфтами контролеру S7-1511-1 PN PLC1-PLC2.

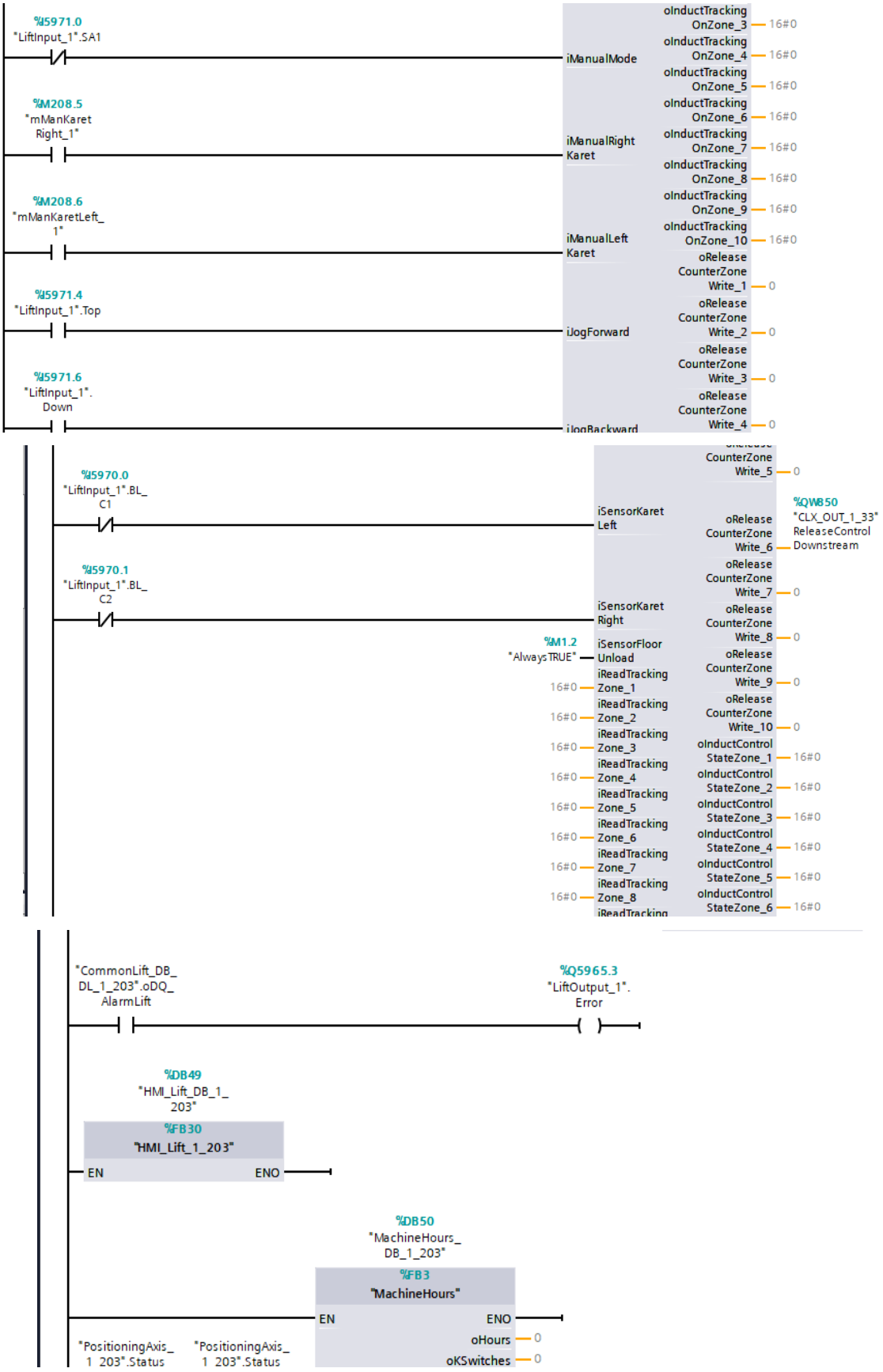


Рисунок 3.18, аркуш 2

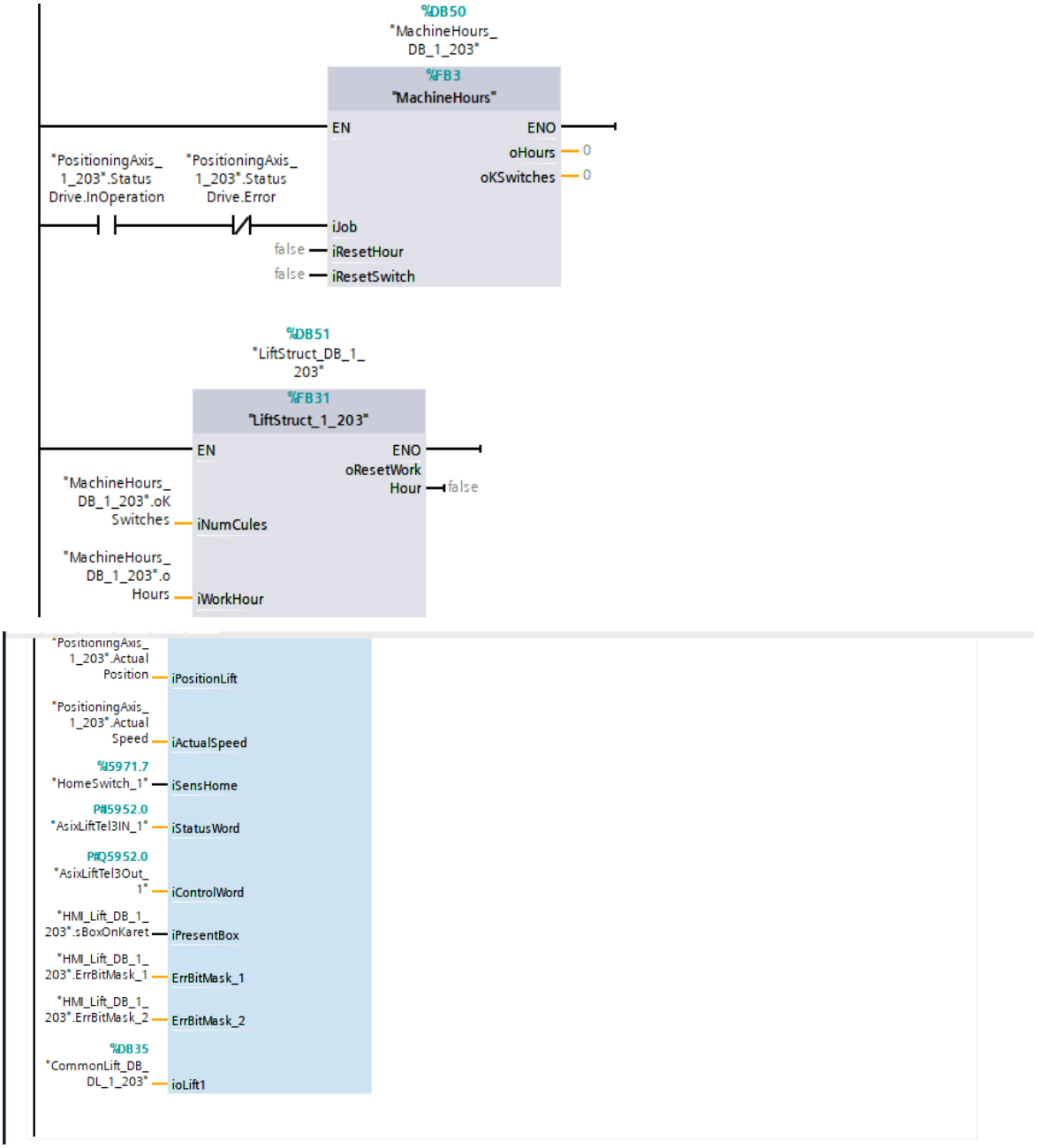


Рисунок 3.18, аркуш 3

Далі потрібно створити найголовніші блоки, це організаційні блоки для керування conveylinx AI2, таких блоків має бути 3 штуки, ці блоки дозволять керувати всіма конвеєрами, вони мають керувати рухом конвеєру, трекінгом положення товару на конвеєрі на рисунку 3.19.

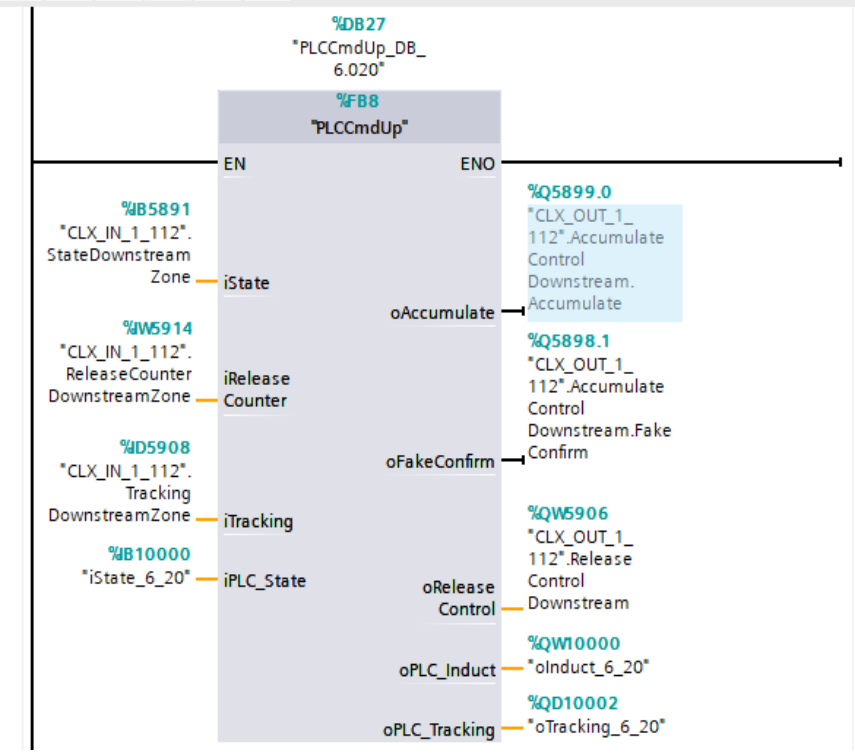


Рисунок 3.19 – Організаційні блоки керування conveylinx AI2 контролеру S7-1511-1 PN PLC1-PLC2.

Далі потрібно створити можливість обміну даними між PLC в системі за допомогою витої пари або інакше промислового стандарту profinet.

```
IF "sstartsystem" AND NOT "globalerror" AND NOT "globalemergencystop" THEN
    "ssystemwork" := TRUE;
END_IF;
IF "sstopsystem" OR "globalerror" OR "globalemergencystop" THEN
    "ssystemwork" := FALSE;
END_IF;

//Пуск системи
"ostart_MSH2" := "ssystemwork";
"ostart_RSH1" := "ssystemwork";
"ostart_RSH2" := "ssystemwork";
"ostart_RSH3" := "ssystemwork";
```

```
"ostart_RSH4" := "ssystemwork";
"ostart_RSH5" := "ssystemwork";
"ostart_PLC2" := "ssystemwork";
"ostart_PLC3" := "ssystemwork";
"ostart_PLC4" := "ssystemwork";
"ostart_PLC5" := "ssystemwork";
"ostart_PLC6" := "ssystemwork";
"ostart_PLC7" := "ssystemwork";
"ostart_PLC8" := "ssystemwork";
"ostart_PLC9" := "ssystemwork";
"ostart_PLC10" := "ssystemwork";

//Аварійниа зупинка в зоні ПЛК4
(* "semergencystop" := "S1_2" OR "S1_3" OR "S1_5" OR "S1_6" OR "S1_7" OR
"S1_8" OR
"S1_9" OR "S1_10" OR "S1_11" OR "S1_12" OR "S1_13" OR "S1_14" OR "S1_15"
OR
"S1_16" OR "S1_17" OR "S1_18" OR "S1_19" OR "S1_20" OR "S1_21" OR "S1_22"
OR
"S1_23" OR "S1_24" OR "S1_25" OR "S1_26" OR "S1_27" OR "S1_28" OR "S1_29"
OR
"S1_30" OR "S1_31" OR "S1_32" OR "S1_33"; *)

//Помилка обладнання в зоні ПЛК4
(* "serrarea" := "Data_beltshort"."THIS"[1].error_object OR
"Data_beltshort"."THIS"[2].error_object OR
"Data_beltshort"."THIS"[3].error_object OR
"Data_TPR"."THIS"[1].error OR "Data_TPR"."THIS"[2].error OR
"Data_TPR"."THIS"[3].error OR "Data_TPR"."THIS"[4].error OR
"Data_TPR"."THIS"[5].error OR "Data_TPR"."THIS"[6].error OR
```

```
"Data_TPR"."THIS"[7].error; *)

//Помилки по зонам
"level_0_error" := "ifault_PLC2" OR "ifault_PLC3" OR "ifault_PLC4" OR "ifault_PLC5";
"level_0_receiving_error" := "ifault_PLC3";
"level_0_sorting_error_1" := "ifault_PLC2" OR "serrarea";
"level_0_sorting_error_2" := "ifault_PLC9";
"level_1_error" := "ifault_PLC4" OR "ifault_PLC5";
"level_2_error" := "ifault_PLC7" OR "ifault_PLC8" OR "ifault_PLC9" OR "ifault_PLC10";
"level_2_sorting" := "ifault_PLC9" OR "ifault_PLC10";
"level_2_storage" := "ifault_PLC7" OR "ifault_PLC8";
"level_2_error" := "ifault_PLC7" OR "ifault_PLC8" OR "ifault_PLC10";
```

Також для пуску був написаний наступний код

```
IF "sstartsystem" AND NOT "globalerror" AND NOT "globalemergencystop" THEN
   "ssystemwork" := TRUE;
END_IF;
IF "sstopsystem" OR "globalerror" OR "globalemergencystop" THEN
   "ssystemwork" := FALSE;
END_IF;

//Пуск системи
"ostart_MSH2" := "ssystemwork";
"ostart_RSH1" := "ssystemwork";
"ostart_RSH2" := "ssystemwork";
"ostart_RSH3" := "ssystemwork";
"ostart_RSH4" := "ssystemwork";
"ostart_RSH5" := "ssystemwork";
```

```
"ostart_PLC2" := "ssystemwork";
"ostart_PLC3" := "ssystemwork";
"ostart_PLC4" := "ssystemwork";
"ostart_PLC5" := "ssystemwork";
"ostart_PLC6" := "ssystemwork";
"ostart_PLC7" := "ssystemwork";
"ostart_PLC8" := "ssystemwork";
"ostart_PLC9" := "ssystemwork";
"ostart_PLC10" := "ssystemwork";

//Аварійний стоп в зоні ПЛК4
(* "semergencystop" := "S1_2" OR "S1_3" OR "S1_5" OR "S1_6" OR "S1_7" OR
"S1_8" OR
"S1_9" OR "S1_10" OR "S1_11" OR "S1_12" OR "S1_13" OR "S1_14" OR "S1_15"
OR
"S1_16" OR "S1_17" OR "S1_18" OR "S1_19" OR "S1_20" OR "S1_21" OR "S1_22"
OR
"S1_23" OR "S1_24" OR "S1_25" OR "S1_26" OR "S1_27" OR "S1_28" OR "S1_29"
OR
"S1_30" OR "S1_31" OR "S1_32" OR "S1_33"; *)

//Ошибка оборудования в зоне ПЛК4
(* "serrarea" := "Data_beltshort"."THIS"[1].error_object OR
"Data_beltshort"."THIS"[2].error_object OR
"Data_beltshort"."THIS"[3].error_object OR
"Data_TPR"."THIS"[1].error OR "Data_TPR"."THIS"[2].error OR
"Data_TPR"."THIS"[3].error OR "Data_TPR"."THIS"[4].error OR
"Data_TPR"."THIS"[5].error OR "Data_TPR"."THIS"[6].error OR
"Data_TPR"."THIS"[7].error; *)
```

```
//Помилка в зонах
"level_0_error" := "ifault_PLC2" OR "ifault_PLC3" OR "ifault_PLC4" OR "ifault_PLC5";
"level_0_receiving_error" := "ifault_PLC3";
"level_0_sorting_error_1" := "ifault_PLC2" OR "serrarea";
"level_0_sorting_error_2" := "ifault_PLC9";
"level_1_error" := "ifault_PLC4" OR "ifault_PLC5";
"level_2_error" := "ifault_PLC7" OR "ifault_PLC8" OR "ifault_PLC9" OR "ifault_PLC10";
"level_2_sorting" := "ifault_PLC9" OR "ifault_PLC10";
"level_2_storage" := "ifault_PLC7" OR "ifault_PLC8";
"level_2_error" := "ifault_PLC7" OR "ifault_PLC8" OR "ifault_PLC10";
```

3.7 Створення функціональних блоків PLC Siemens S7-1511-1 PN PLC1 – PLC2

Важливими функціональними блоками є блок рахунку машинних годин, який дозволить вести рахунок, та в потрібний момент повідомити що той чи інший вузол наближається до кінця свого робочого циклу. Він виглядає наступним чином:

```
#R_TRIG_Clock(CLK:="Clock_1Hz");
#R_TRIG_Switch(CLK := #ijob);
IF #ijob AND #R_TRIG_Clock.Q THEN
   #ssecond := #ssecond + 1;
   IF #ssecond >= 3600 THEN
      #ssecond := 0;
      #ohours := #ohours + 1;
   END_IF;
END_IF;
```

```
IF #R_TRIG_Switch.Q THEN
   #sswitches := #sswitches + 1;
   IF #ssecond >= 1000 THEN
      #sswitches := 0;
      #okswitches := #okswitches + 1;
   END_IF;
END_IF;
IF #iresethour THEN
   #ssecond := 0;
   #ohours := 0;
END_IF;
IF #iresetswitch THEN
   #sswitches := 0;
   #okswitches := 0;
END_IF;
```

Код, що наведений, реалізовано на мові програмування для контролерів Siemens S7-1511 у середовищі розробки TIA Portal. Він виконує функцію підрахунку часу і кількості вимикачів. Використовуючи перехідні блоки для визначення моменту часу та активації вимикача. Після кожного циклу перевіряється, чи досягнуто певних меж для скидання часу або вимикача і збільшення значення годин. Таким чином, код допомагає вести облік часу та кількості вимикачів у системі.

Блок керування Атопом:

```
#oaccumulatezone := TRUE;//Акумуляція зони
//Якщо ящик з попереднюї зони випускається і наступна зона його приймає
//Тоді ящик зупиняється і чекає, коли його приберуть
IF #istatezona = 2 THEN
   #releasebox := FALSE;
```

```
ELSIF #istatezona = 1 THEN
   #releasebox := TRUE;
END_IF;
//Затримка часу
#timedelaybox(IN := #istatezona = 5,
            PT := #itimedelay);
//Умова випуску ящика з зони
IF #timedelaybox.Q AND #releasebox THEN
   #oreleasezone := #ireleasezone + 1;
END_IF;
```

Активуючи акумуляцію зони, встановлюється змінна #oaccumulatezone в TRUE. Далі, перевіряється стан зони: якщо ящик переміщується з попередньої зони до наступної, встановлюється #releasebox в FALSE, щоб зупинити його рух. Якщо ящик прибирають і наступна зона його приймає, встановлюється #releasebox в TRUE, щоб дозволити його рух.

Далі, встановлюється затримка часу для ящика за допомогою функції #timedelaybox. Це означає, що ящик буде затриманий протягом певного часу (#itimedelay), якщо стан зони дорівнює 5.

Нарешті, перевіряється умова для випуску ящика з зони: якщо затримка часу завершена і #releasebox встановлено в TRUE, то збільшується значення #oreleasezone (це, ймовірно, лічильник випущених ящиків з зони).

Керування стрічковим конвеєром:

```
//Акумуляція зони перед стрічковим ковеєром
#oacumulateback := TRUE;
#tonrun := FALSE;
#oerrorconveybelt := FALSE;
CASE #State OF
   0:
```

```
        //Ініціалізація конвеєра
        #orunbeltconvey := TRUE;
        #State := 100;
    100:
        //Очікування ящику на зоні перед ліфтом
        #tonrun := TRUE; //Закпуск таймера контролю випуску ящика з ковеєра
        IF (#istateback = 5 OR #istateback = 2) AND (NOT #isensorfull (*OR #isensorfull AND #orunbeltconvey *)) THEN
            //Чекання лотка на зоні перед стрічковим ковеєром
            #orunbeltconvey := TRUE;
            #tonrun := FALSE;
            IF #delaystartzone.Q AND #istateback = 5 THEN
                //Випуск дотка з затрикою часу, для  запуску стрічки.
                #oreleaseback := #ireleaseback + 1;
            END_IF;
        ELSIF #isensorfull AND (#istatenext = 1 OR #istatenext = 2) THEN
            //Якщо є sensorfull  на стрічковому конвеєрі запускаєтья лента
            #orunbeltconvey := TRUE;
        END_IF;
        IF #isensorfull AND (#istatenext = 5) OR #delayinitconvey.Q THEN
            //Зупика конвеєра якщо спрацював таймер або наступна зона за конвеєром зайнята і спрацював sensorfull
            #orunbeltconvey := FALSE;
            #tonrun := FALSE;

        END_IF;

    66:
        #orunbeltconvey := FALSE;
        #oerrorconveybelt := TRUE;
```

```
            IF #ireseterror THEN
               //Скидання помилки
               #State := 0;
            END_IF;
         END_CASE;
         //Таймер для затрики випуску лотка на стрічку
         #delaystartzone(IN := #orunbeltconvey,
                         PT := T#600ms);
         //Таймер контролю ркуху стрічки
         #delayinitconvey(IN := #tonrun AND #orunbeltconvey AND NOT #isensorfull,
                          PT := #itimeinitconvey);
         IF #istateback <> 4 AND #istateback <> 5 THEN
            //Умово  запуску таймера  для Fake
            #tpfake := TRUE;
         ELSE
            #tpfake := FALSE;
         END_IF;
         //Таймер для запуску біта
         #delayfakeconfigbit(IN := #tpfake,
                             PT := T#200ms);
         #ofakeconfig := #delayfakeconfigbit.Q;
         #sfc_Control(HW_Identifier := #ifc_Identifier,
                      FI_Systembus := 1,
                      Enable:= (#iautomode AND #orunbeltconvey) OR (NOT #iautomode
AND #irunmanual) OR #sauto,
                      Disable_Voltage := false,
                      Quick_Stop := #iemergencystop,
                      Fault_ACK:=#ireseterror OR #sreseterror,
                      Direction := #idirection,
                      Parameter_Set:=1,
```

```
            Setpoint_1:=#isetspeed*10
            );
        IF #sfc_Control.Fault THEN
            //Зовнішння помилка
            #oerrorconveybelt := TRUE;
            #orunbeltconvey := FALSE;
            #State := 66;
        END_IF;
        IF #ilandslidebelt THEN
            //Схід лети
            #oerrorconveybelt := TRUE;
            #orunbeltconvey := FALSE;
            #State := 66;
        END_IF;
        //Встановлення статусу Ready
        IF #State = 100 THEN
            #statusready := TRUE;
        ELSE
            #statusready := FALSE;
        END_IF;
        #machinehours_Instance(ijob:=#orunbeltconvey,
                               Iresethour:=#iresetworkhourmachine OR #sresetmotorhour,
                               Ohours=>#oworkhourmachine);
        REGION Data
            //Передаються постійно
            #iodatabeltshort[#inumbelt].releasebox := #oreleaseback;
            #iodatabeltshort[#inumbelt].running := #orunbeltconvey;
            #iodatabeltshort[#inumbelt].error_object := #oerrorconveybelt OR
#sfc_Control.Not_Ready_to_Switch_On;
            #iodatabeltshort[#inumbelt].error_drive := #sfc_Control.Fault;
```

```
        // Передається лише при запиті з HMI
        IF #inumbelt = #inumbeltfromhmi THEN
            //Status Zone
            #iodatabeltfull[#inumbelt].statuspre := #istateback;
            #iodatabeltfull[#inumbelt].statusafter := #istatenext;
            #iodatabeltfull[#inumbelt].releasebox := #oreleaseback;
            #iodatabeltfull[#inumbelt].sensorfull := #isensorfull AND #istatenext = 5;
            //Status Object
            //Тут має бути акутальна швидкість, після переведення з RPM
            #iodatabeltfull[#inumbelt].motorhours := #oworkhourmachine;
            #sresetmotorhour :=  #iodatabeltfull[#inumbelt].resetmotorhours;
            #iodatabeltfull[#inumbelt].numstateac := #State;
            #iodatabeltfull[#inumbelt].running := #orunbeltconvey;
            #iodatabeltfull[#inumbelt].alarm := #oerrorconveybelt OR
#sfc_Control.Not_Ready_to_Switch_On;
            #iodatabeltfull[#inumbelt].error := #oerrorconveybelt OR
#sfc_Control.Fault OR #sfc_Control.Not_Ready_to_Switch_On;
            #sreseterror := #iodatabeltfull[#inumbelt].resetalarmerror;
            #sauto := #iodatabeltfull[#inumbelt].automode;
            #srun := #iodatabeltfull[#inumbelt].on;
            #iodatabeltfull[#inumbelt].ready := #statusready AND
#sfc_Control.Ready_to_Switch_On;
            //Drive
            #iodatabeltfull[#inumbelt].error_drive := #sfc_Control.Fault;
            #iodatabeltfull[#inumbelt].numerror := #sfc_Control.Actual_Value_2;
            #iodatabeltfull[#inumbelt].actualspeed := #sfc_Control.Actual_Value_1;
            #iodatabeltfull[#inumbelt].current := #sfc_Control.Actual_Value_3 / 1000;
            //#iodatabeltfull.actualrpm := ABS(REAL_TO_INT(#Index[4].srvalue));
            //#iodatabeltfull.actualspeed := #iodatabeltfull.actualrpm / 800;
            //#iodatabeltfull.actualpower :=  REAL_TO_INT(#Index[5].srvalue);
```

```
            //statusworddrive
            //#iodatabeltfull.nospeeddeviation := #itelegramin.ZSW1.nospeeddeviation;
            //#iodatabeltfull.controlrequested := #itelegramin.ZSW1.controlrequested;
            //#iodatabeltfull.speedcomparisonvalusreachedexeeded :=
#itelegramin.ZSW1.speedcomparisonvalusreachedexeeded;
            //#iodatabeltfull.torquelimitnotreached :=
#itelegramin.ZSW1.torquelimitnotreached;
            //#iodatabeltfull.openholdingbrake :=
#itelegramin.ZSW1.openholdingbrake;
            //#iodatabeltfull.nomotorovertemperature :=
#itelegramin.ZSW1.nomotorovertemperature;
            #iodatabeltfull[#inumbelt].readytoswitchon :=
#sfc_Control.Not_Ready_to_Switch_On;
            #iodatabeltfull[#inumbelt].readytooperate :=
#sfc_Control.Ready_to_Switch_On;
            #iodatabeltfull[#inumbelt].operationenabled :=
#sfc_Control.Operation_Enabled;
            #iodatabeltfull[#inumbelt].faultpresent := #sfc_Control.Fault_Active;
            //#iodatabeltfull.nocoaststopactivated :=
#itelegramin.ZSW1.nocoaststopactivated;
            #iodatabeltfull[#inumbelt].noquickstopactivated :=
#sfc_Control.Quick_Stop_Active;
            #iodatabeltfull[#inumbelt].switchingonnotinhibited :=
#sfc_Control.Switch_On_Inhibit;
            //#iodatabeltfull.alarmpresent := #itelegramin.ZSW1.alarmpresent;
            //controlword
            (*#iodatabeltfull.controlbyplc := #itelegramout.STW1.controlbyplc;
            #iodatabeltfull.setpointinversion := #itelegramout.STW1.setpointinversion;
            #iodatabeltfull.openholdingbrake_1 :=
#itelegramout.STW1.openholdingbrake;
```

```
            #iodatabeltfull.lowermotorizedpotentiometersetpoint :=
#itelegramout.STW1.lowermotorizedpotentiometersetpoint;
            #iodatabeltfull.on := #itelegramout.STW1.On;
            #iodatabeltfull.nocoaststop := #itelegramout.STW1.nocoaststop;
            #iodatabeltfull.noquickstop := #itelegramout.STW1.nocoaststop;
            #iodatabeltfull.enableoperation := #itelegramout.STW1.enableoperation;
            #iodatabeltfull.enablerampgenerator :=
#itelegramout.STW1.enablerampgenerator;
            #iodatabeltfull.enablesetpoint := #itelegramout.STW1.enablesetpoint;
            #iodatabeltfull.faultacknowledge :=
#itelegramout.STW1.faultacknowledge;*)
            // Error
            (*#iodatabeltfull.circuitbreaker := #imaincircuitbreaker;
            #iodatabeltfull.maincontaktor := #imaincontaktor;
            #iodatabeltfull.driveerror := #itelegramin.ZSW1.alarmpresent;*)
            //Alarm
            #iodatabeltfull[#inumbelt].emergencystop := #iemergencystop;
          ELSE
            #iodatabeltfull[#inumbelt] := #reserallstructfull;
          END_IF;
        END_REGION
```

Цей код відображає програму керування стрічковим конвеєром з використанням різних датчиків та таймерів для ефективної роботи та контролю за процесом. Активуючи акумуляцію зони перед стрічковим конвеєром, встановлюється змінна #oacumulatezone в TRUE. Оброблюються різні стани зони за допомогою оператора CASE, включаючи ініціалізацію конвеєра та чекання ящика на зоні перед ліфтом. Використовуються таймери для затримки старту зони та контролю руху ленти. Керування приводом стрічкового конвеєра здійснюється за допомогою функції #sfc_Control, а також обробляються різні ситуації, пов'язані

з помилками та аваріями. Дані про роботу конвеєра передаються для моніторингу та аналізу.[9]

Блок автоматичної роботи ліфту:

```
REGION скидання вихідних признаків
   #ozproc := false;
   #onproc := false;
   #ozinit := false;
   #oninit := false;
   #ozerr := false;
   #onerr := false;
   #oinitunload := false;
   #odonelift := false;
   #ozinitpositionlift := false;
   #oninitpositionlift := false;
   #oreleasedownload := false;
   #oreleaseunload := false;
   #ozrunkaretdownload := false;
   #onrunkaret := false;
   #ozrunkaretunload := false;
   #ozunloadspeed := false;
   #ozdownloadspeed := false;
   #onunloadspeed := false;
   #ondownloadspeed := false;
   #ozunload := false;
   #ozdownload := false;
   #onunload:= false;
   #ondownload := false;
   #ostartmotioncontrol := false;
   //определение кол-ва уставок
```

```
    #countarray_ztacs(Array:=#CTRL.oarrzstacs);
    #countarray_zdelay(Array := #CTRL.oarrzdelay);
    FOR #i := 0 TO 8 DO
        #CTRL.oarrzstacs[#i] := false;
    END_FOR;
    FOR #i := 0 TO 8 DO
        #CTRL.oarrzdelay[#i] := false;
    END_FOR;
END_REGION
// Перетин двох сенсорів негабаритний товар
IF #isenskaretfirst AND #isenskaretsecond THEN
    #ERR.arrerrcode[1] := true;
    #snumstate := 66;
END_IF;
// Зовнішня помилка
IF #ierr THEN
    #snumstate := 66;
END_IF;
IF NOT #iautomode  THEN
    #snumstate := 66;
END_IF;
#R_TRIG_Instance(CLK := #iautomode);
IF #R_TRIG_Instance.Q THEN
    #snumstate := 0;
END_IF;
// IF #ikreserr THEN
//     #snumstate := 0;
//     #onerr := true;
// END_IF;
CASE #snumstate OF
```

```
0://Ініціалізація
   FOR #i := 1 TO 20 DO
      #ERR.arrerrcode[#i] := false;
   END_FOR;
      #snumstate := 100;
      #ozinit := true;
      #sinitposkaretka := FALSE;
      #sprochomepos := false;
100:  //Перевірка положення каретки
   IF  #isenskaretfirst THEN
      #ozrunkaretdownload := true;
      #snumstate := 101;
   ELSIF #isenskaretsecond THEN
      #ozrunkaretunload := true;
      #snumstate := 111;
   ELSE
      #snumstate := 102;
   END_IF;
101: // Рух каретки до перетину датчика isenskaretfirst
   #CTRL.oarrzstacs[0] := true;
   IF #FDBK.iarrtacs[0]  THEN
      #ERR.arrerrcode[1] := 1;
      #snumstate := 66;
      ELSIF NOT #isensorkaretleft AND NOT #isensorkaretright  THEN
      #snumstate := м2;
   END_IF;
111: // Рух каретки до перетену датчика isenskaretsecond
   #CTRL.oarrzstacs[0] := true;
   IF #FDBK.iarrtacs[0] THEN
      #ERR.arrerrcode[1] := 1;
```

```
        #snumstate := 66;
      ELSIF NOT #isensorkaretleft AND NOT #isensorkaretright THEN
        #snumstate := 112;
      END_IF;
   112:  //Затримка після перетину датчика каретки та контроль сенсорів каретки
      #CTRL.oarrzdelay[1] := true;
      IF #FDBK.iarrtdelay[1] AND NOT #isenskaretsecond AND NOT #isenskaretfirst
THEN
        #onrunkaret := true;
        #snumstate := 102;
      ELSIF #isensorkaretleft OR #isensorkaretright THEN
        #ERR.arrerrcode[18] := 1;
        #snumstate := 66;
      END_IF;
   102: //Задержка перед стартом иниц. Каретки
      #CTRL.oarrzdelay[6] := true;
      IF #FDBK.iarrtdelay[6] AND NOT #isenskaretsecond AND NOT #isenskaretfirst
THEN
        #ostartmotioncontrol := true;
        #snumstate := 103;
      ELSIF  #isensorkaretleft OR #isensorkaretright THEN
        #ERR.arrerrcode[18] := 1;
        #snumstate := 66;
      END_IF;
   103: //Позіціонування в home position
      #CTRL.oarrzstacs[8] := true;
      IF #FDBK.iarrtacs[8] OR #ikreserr THEN
        #ERR.arrerrcode[20] := true;
        #snumstate := 66;
      ELSIF #ihomepositiondone THEN
```

```
        #snumstate := 1;
    END_IF;
1: //Перевірка положения каретки при старті
    IF #isenskaretfirst OR #isenskaretsecond THEN
        #ERR.arrerrcode[18] := true;
        #snumstate := 66;
    ELSIF #isensunloadfloor AND NOT #sprochomepos THEN
        #oinitunload := true;
        #snumstate := 3;
    ELSE
        #ozunloadspeed := true;
        #oinitunload := true;
        #snumstate := 2;
    END_IF;
2: //Рух на поверх розвантаження
    #CTRL.oarrzstacs[1] := true;
    IF #FDBK.iarrtacs[1] THEN
        #ERR.arrerrcode[2] := true;
        #snumstate := 66;
    ELSIF #isenskaretfirst OR #isenskaretsecond THEN
        #ERR.arrerrcode[18] := true;
        #snumstate := 66;
    ELSIF #isenslimit THEN
        #ERR.arrerrcode[11] := true;
        #snumstate := 66;
    ELSIF #isensunloadfloor THEN
        #onunloadspeed := true;
        #snumstate := 3;
    END_IF;
3: //Каретка на поверсі вивантаження
```

```
        #CTRL.oarrzdelay[2] := true;
        #ozinitpositionlift := true;
        IF #FDBK.iarrtdelay[2] AND NOT #isensunloadfloor THEN
            #ERR.arrerrcode[4] := true;
            #snumstate := 66;
        ELSIF  #sprochomepos THEN
            #sprochomepos := false;
            #snumstate := 8;
        ELSIF #FDBK.iarrtdelay[2] AND #iinitpositionlift AND #isensunloadfloor AND
#istatezoneunload1 AND #isensorfloorunload THEN
            #ozrunkaretunload := true;
            #oreleaseunload := true;
            #snumstate := 4;
        END_IF;
    4: //Вивантаження лотка з каретки до спрацювання датчика isenskaretfirst
        #CTRL.oarrzdelay[0] := true;
        IF NOT #isensunloadfloor THEN
            #ERR.arrerrcode[5] := true;
            #snumstate := 66;
        ELSIF #FDBK.iarrtdelay[0] AND #sinitposkaretka THEN
            #ERR.arrerrcode[19] := 1;
            #snumstate := 66;
        ELSIF #FDBK.iarrtdelay[0] AND NOT #sinitposkaretka THEN
            #sinitposkaretka := 1;
            #snumstate := 6;
        ELSIF #isenskaretfirst THEN
            #snumstate := 5;
        END_IF;
    5: //Розвантадення з каретки до відключення датчика isenskaretfirst
        #CTRL.oarrzstacs[3] := true;
```

```
        IF #FDBK.iarrtacs[3] THEN
            #ERR.arrerrcode[6] := true;
            #snumstate := 66;
        ELSIF #istatezoneunload5 THEN
            #oreleaseunload := true;
            #snumstate := 6;
        END_IF;
    6: //Затримка часу після відключення каретки при вигрузці лотка
        #CTRL.oarrzdelay[3] := true;
        IF #FDBK.iarrtdelay[3] AND (#isenskaretsecond OR #isenskaretfirst) THEN
            #ERR.arrerrcode[7] := true;
            #snumstate := 66;
        ELSIF #FDBK.iarrtdelay[3] THEN
            #onrunkaret := true;
            #snumstate := 7;
        END_IF;
    7: //Затримка часу після процесу вивантаження лотка з каретки
        #CTRL.oarrzdelay[4] := true;
        IF #FDBK.iarrtdelay[4] AND (#isenskaretfirst OR #isenskaretsecond) THEN
            #ERR.arrerrcode[8] := true;
            #snumstate := 66;
        ELSIF #FDBK.iarrtdelay[4] AND NOT #isenskaretfirst AND NOT
#isenskaretsecond THEN
            #onunload := true;
            #onproc:= true;
            #oninit:= true;
            #odonelift := true;
            #snumstate := 8;
            RETURN;
        END_IF;
```

```
8: //Очикування команди прибуття лотка
   IF #istartcyclehoming THEN
      #sprochomepos := true;
      #ostartmotioncontrol := true;
      #snumstate := 82;
   ELSIF #isenskaretsecond OR #isenskaretfirst THEN
      #ERR.arrerrcode[9] := true;
      #snumstate := 66;
   ELSIF #imoveliftdownload THEN
      #onunload := true;
      #ozdownload := true;
      #snumstate := 81;
   END_IF;
81: //Якщо лоток прибув
   #CTRL.oarrzdelay[2] := true;
   IF #FDBK.iarrtdelay[2] AND #imoveliftdownload AND #isensdownloadfloor THEN
      #snumstate := 11;
   ELSIF #FDBK.iarrtdelay[2] AND #imoveliftdownload AND NOT #isenskaretfirst AND NOT #isenskaretsecond THEN
      #ozdownloadspeed := true;
      #snumstate := 9;
   END_IF;
82: // Виконання процедури homig придосягенні максимальної кількості циклів для виконнаня homing
   #CTRL.oarrzstacs[8] := true;
   IF #FDBK.iarrtacs[8] THEN
      #ERR.arrerrcode[20] := true;
      #snumstate := 66;
   ELSIF #ihomepositiondone THEN
```

```
      #snumstate := 1;
   END_IF;
9:  // Затрима часу після початку руху до датчику сповільнення
   #CTRL.oarrzdelay[8] := true;
   IF #FDBK.iarrtdelay[8] THEN
      #snumstate := 10;
   END_IF;
10: //Рух до датчику зупинка на поверсі завантаження
   #CTRL.oarrzstacs[5] := true;
   #ozproc := true;
   IF #FDBK.iarrtacs[5] THEN
      #ERR.arrerrcode[12] := true;
      #snumstate := 66;
   ELSIF #isenskaretfirst OR #isenskaretsecond THEN
      #ERR.arrerrcode[13] := true;
      #snumstate := 66;
   ELSIF #isensdownloadfloor THEN
      #ondownloadspeed := true;
      #snumstate := 11;
   END_IF;
11: //Затримка часу після прибуття каретки на поверх завантаження
   #CTRL.oarrzdelay[5] := true;
   #ozproc := true;
   IF #FDBK.iarrtdelay[5] AND NOT #isensdownloadfloor THEN
      #ERR.arrerrcode[14] := true;
      #snumstate := 66;
   ELSIF #FDBK.iarrtdelay[5] AND #isensdownloadfloor THEN
      #ozrunkaretdownload := true;
      #oreleasedownload := true;
      #snumstate := 12;
```

```
    END_IF;
```

12: //Завантаження лотка на каретку ліфта до спрацювання датчика каретки #isenskaretfirst

```
    #CTRL.oarrzstacs[6] := true;
    IF #FDBK.iarrtacs[6] THEN
        #ERR.arrerrcode[15] := true;
        #snumstate := 66;
    ELSIF #isenskaretfirst  THEN
        #snumstate := 13;
    END_IF;
```

13: //Завантаження лотка на каретку ліфту до відключення датчика каретки #isenskaretfirst

```
    #CTRL.oarrzstacs[7] := true;
    #CTRL.oarrzdelay[2] := true;
    IF #FDBK.iarrtacs[7] THEN
        #ERR.arrerrcode[16] := true;
        #snumstate := 66;
    ELSIF NOT #isenskaretfirst AND #FDBK.iarrtdelay[2] THEN
        #onrunkaret := true;
        #snumstate := 14;
    END_IF;
```

14: //Затримка після відключення датчика картки #isenskaretfirst при завантаженні

```
    #CTRL.oarrzdelay[6] := true;
    #ondownload := true;
    IF #FDBK.iarrtdelay[6] THEN
        #snumstate := 15;
    END_IF;
```

15: //Затримка після завантадення лотка на каретку

```
    #CTRL.oarrzdelay[7] := true;
```

```
   #ozunload := true;
   IF #isenskaretfirst OR #isenskaretsecond THEN
      #ERR.arrerrcode[17] := true;
      #snumstate := 66;
   ELSIF #FDBK.iarrtdelay[7] THEN
      #ozunloadspeed := true;
      #snumstate := 16;
   (*ELSIF #isensunloadfloor THEN
      #snumstate := 3;*)   /// якщо поверх завантаження = поверху вивантаження
   END_IF;

16: // Затримка часу після початку руху до датчику
   #CTRL.oarrzdelay[8] := true;
   IF #FDBK.iarrtdelay[8] THEN
      #snumstate := 2;
   END_IF;
66:  //Стан аварії
   #onproc := TRUE;
   #oninit := TRUE;
   #ozerr := TRUE;
   #oninitpositionlift:= TRUE;
   #onrunkaret := TRUE;
   #onunload := true;
   #onunloadspeed := TRUE;
   #ondownloadspeed := TRUE;
   #ostartmotioncontrol := FALSE;
   #sprochomepos := false;
   IF #ikreserr THEN
      #snumstate := 0;
      #onerr := true;
```

```
    END_IF;

  ELSE
    #snumstate := 66;

END_CASE;
```

Наведений код представляє програму керування роботою ліфта на стрічковому конвеєрі, яка забезпечує ефективну та безпечну роботу системи. Під час виконання програма використовує різні датчики та таймери для контролю різних аспектів процесу.

Починаючи з обнулення вихідних сигналів, програма забезпечує чистий початок роботи системи, готуючи її до наступних етапів. Далі відбувається перевірка стану датчиків та реагування на різні ситуації, які можуть виникнути під час роботи.

Основна функціональність розподілена на різні стани (state), які визначають поведінку системи на кожному етапі роботи. Це дозволяє чітко структурувати програму та легко розширювати або модифікувати її в майбутньому. Використання таймерів дозволяє синхронізувати рух ліфта та контролювати тривалість певних операцій. Це важливо для забезпечення безпеки та ефективності роботи системи. Крім того, програма передбачає обробку аварійних ситуацій, які можуть виникнути під час роботи ліфта. Вона автоматично реагує на ці ситуації та приймає відповідні заходи для забезпечення безпеки та подальшої нормальної роботи системи. Узагальнюючи, код програми керування роботою ліфта на стрічковому конвеєрі є важливим елементом для забезпечення безперебійної та ефективної роботи ліфтової системи. Він використовує різні технології та методи для контролю та керування різними аспектами процесу.

3.8 Тестування працездатності та коректної роботи контролеру Siemens S7-1511-1 PN PLC1 – PLC2

Тестування було проведено так само як і для контролерів S7-1511-1 PN PLC1 – PLC2. Правильне тестування з використанням PLC Simulator у Siemens TIA Portal є критично важливим етапом у розробці автоматизованих систем. Перш ніж розпочати, необхідно створити проект у TIA Portal та налаштувати його для конкретного PLC. Напишіть програму у мові програмування, яка підтримується TIA Portal, таку як Ladder Logic або Structured Text, забезпечивши її відповідністю вимогам проекту. Після цього PLC Simulator у проекті та запустіть симуляцію. Це дозволить відтворити роботу програми у віртуальному середовищі. Підключіться до PLC Simulator через TIA Portal, щоб моніторити роботу програми та взаємодіяти з нею у реальному часі. Ретельне тестування програми, всі її функції та взаємодію з введеними даними. Виявлені помилки виправляються та повторюється тестування. Після успішного завершення тестування здійснюється валідація та оптимізацію програми, враховуючи будь-які виявлені під час тестування недоліки чи можливі покращення. Такий підхід допоможе забезпечити надійність та ефективність вашої автоматизованої системи перед її впровадженням у реальне середовище.

Ці ключові перевірки дозволили зробити систему надійною, розсортувати помилки та аварії по рівням їх критично.

4. РОЗРАХУНОК НАДІЙНОСТІ СИСТЕМИ

Надійність автоматизованої системи управління конвеєрною лінією (АСУКЛ) визначається її здатністю безвідмовно виконувати свої функції протягом заданого часу. Це особливо важливо для промислових застосувань, де відмови можуть призводити до значних втрат продуктивності та фінансових збитків. Розрахунки надійності дозволяють визначити і мінімізувати ризики, забезпечуючи безперебійну роботу системи. Враховуючи складність сучасних АСУКЛ, що включають різноманітні компоненти, такі як контролери, датчики, мотори та програмне забезпечення, необхідно ретельно аналізувати їх взаємодію та ймовірність відмови кожного елемента. В даній роботі розглянемо методи розрахунку надійності для основних компонентів системи та оцінку загальної надійності системи в цілому. Це дозволить запропонувати заходи для підвищення надійності та ефективності роботи АСУКЛ, забезпечуючи стабільну і продуктивну експлуатацію у промислових умовах.

Постановка задачі:

Розрахувати надійність АСУКЛ, що складається з наступних компонентів:

– контролери Siemens S7-1200 та S7-1500;

– оптичні датчики SICK;

– сканер штрих кодів SICK;

– мотор NordCon 8Кв.;

– програмне забезпечення Siemens.

Для розрахунку надійності використовуються наступні моделі:

– експоненціальна модель для безвідмовної роботи компонентів;

– системи з резервуванням.

Початкові дані:

Припустимо, що середній час безвідмовної роботи (MTBF) для кожного компонента визначено експериментально:

– контролер Siemens S7-1200: MTBF = 20000 годин;

– контролер Siemens S7-1500: MTBF = 25000 годин;

– датчики: MTBF = 15000 годин;

– мотори: MTBF = 10000 годин.

Розрахунки за допомогою експоненціальної моделі:

Надійність $R(t)$ системи визначається за формулою:

$$R(t) = e^{-\gamma t}$$

де $\gamma = \frac{1}{MTBF}$;

Для кожного компонента:

– контролер Siemens S7-1200: $\gamma_{S7-1200} = \frac{1}{20000} = 0{,}00005$;

– контролер Siemens S7-1500: $\gamma_{S7-1500} = \frac{1}{25000} = 0{,}00004$;

– датчики SICK: $\gamma_{Sensor} = \frac{1}{15000} = 0{,}0000667$;

– мотор Nordcon 8Кв.: $\gamma_{Motor} = \frac{1}{10000} = 0{,}0001$.

Система без резервування:

Загальна інтенсивність відмов системи : γ_{System}.

$$\gamma_{System} = \gamma_{S7-1200} + \gamma_{S7-1500} + \gamma_{Sensor} + \gamma_{Motor}$$

$$\gamma_{System} = 0{,}00005 + 0{,}00004 + 0{,}0000667 + 0{,}0001 = 0{,}0002567$$

Надійність системи за час t:

$$R_{system}(t) = e^{-0{,}0002567t};$$

Наприклад, для часу $t = 1000$ годин:

$$R_{system}(1000) = e^{-0{,}2567} \approx 0{,}773.$$

Для розрахунку надійності системи використано експоненціальну модель, оскільки вона є простою та ефективною для визначення надійності компонентів з постійною інтенсивністю відмов. Ця модель базується на припущенні, що інтенсивність відмов компонента не змінюється з часом, що добре підходить для більшості електронних і механічних систем на ранніх стадіях їх експлуатації. Інтенсивність відмов γ обчислюється як обернена величина до середнього часу безвідмовної роботи (MTBF). Це дозволяє визначити ймовірність відмови компонента в одиницю часу. Сумування інтенсивностей відмов окремих компонентів дозволяє знайти загальну інтенсивність відмов системи γ_{system}.

Отримавши загальну інтенсивність відмов, можна обчислити надійність системи за допомогою експоненціальної функції $R(t) = e^{-\gamma_{system}t}$ Це дає змогу визначити ймовірність безвідмовної роботи системи протягом заданого періоду часу.

Система з резервуванням:

Якщо в системі додати резервування компонентів, то надійність визначається іншими методами, зокрема методом біноміальних коефіцієнтів або моделюванням системи з резервуванням.

Розрахунки показують, що для заданої системи, що складається з контролерів Siemens, датчиків і моторів, надійність системи без резервування через 1000 годин становить приблизно 77.3%. Збільшення надійності може бути досягнуто шляхом впровадження резервування критичних компонентів. Також слід проводити регулярне тестування та обслуговування системи для забезпечення її стабільної роботи.

5 ОХОРОНА ПРАЦІ

5.1 Охорона праці при автоматизації конвеєрних ліній

Охорона праці є важливим аспектом будь-якої виробничої діяльності, спрямованої на забезпечення безпечних умов праці, захисту здоров'я та життя працівників, а також на запобігання травматизму та професійним захворюванням. При розробці програмного забезпечення для систем автоматизації конвеєрної лінії необхідно враховувати всі аспекти охорони праці, щоб забезпечити безпечну та ефективну експлуатацію обладнання.

5.2Аналіз небезпек і шкідливих факторів

Робота з автоматизованими конвеєрними лініями пов'язана з низкою потенційних небезпек та шкідливих факторів, серед яких механічні небезпеки, електричні небезпеки та шкідливі фактори. Механічні небезпеки включають рухомі частини конвеєра, які можуть призвести до травм працівників у разі недотримання правил безпеки, зокрема затягування одягу чи кінцівок у приводні механізми. Швидкість руху конвеєра може досягати 1 м/с, що підвищує ризик травмування. Електричні небезпеки виникають під час роботи з обладнанням, що працює під напругою до 380 В. Відсутність належного заземлення може призвести до пробоїв і пожеж. Шкідливі фактори включають шум на робочих місцях (до 85 дБА), вібрацію від обладнання (понад 2 м/с²) та випромінювання від електронного обладнання (до 3 В/м).

5.3 Заходи з охорони праці

Для забезпечення безпечних умов праці під час роботи з конвеєрними системами та програмним забезпеченням необхідно дотримуватися інженерно-технічних, організаційних заходів та використовувати індивідуальні засоби

захисту. Інженерно-технічні заходи включають встановлення захисних огорож навколо рухомих частин конвеєра, використання аварійних вимикачів для швидкого зупинення конвеєра у разі небезпеки та забезпечення належного заземлення електрообладнання. Організаційні заходи передбачають регулярне проведення інструктажів з охорони праці та навчання персоналу безпечним методам роботи, розробку та впровадження інструкцій з охорони праці, а також проведення регулярних перевірок стану обладнання та засобів захисту. Індивідуальні засоби захисту включають використання працівниками захисних окулярів, рукавичок, касок, засобів захисту слуху та спеціального робочого одягу.

5.4 Пожежна безпека

У зв'язку з можливістю виникнення пожеж у місцях встановлення електричного та електронного обладнання, необхідно встановити системи автоматичного пожежогасіння та сигналізації, забезпечити наявність вогнегасників у безпосередній близькості до обладнання та проводити регулярні навчання персоналу діям у разі виникнення пожежі.

5.5 Висновки

Дотримання вимог охорони праці під час розробки та експлуатації автоматизованих конвеєрних систем є необхідною умовою для забезпечення безпечних умов праці та зниження ризиків виникнення травматизму і професійних захворювань. Розроблене програмне забезпечення повинно враховувати ці вимоги, забезпечуючи надійне керування системою та можливість швидкого реагування у разі виникнення небезпеки.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи на кафедрі комп'ютерно-інтегрованих технологій, автоматизації та робототехніки Харківського національного університету радіоелектроніки була проведена наукова діяльність, спрямована на розробку та впровадження конвеєрної системи.

За результатами роботи було обрано відповідне обладнання для конвеєрної системи, зокрема роликові та вертикальні конвеєрні секції, а також датчики та пристрої керування. Використовуючи середовище розробки програмного забезпечення TIA Portal, було розроблено програмне забезпечення, яке керує роботою системи.

Програмне забезпечення було розроблене з урахуванням специфіки конвеєрної системи та вимог до її роботи. Це включало в себе програмування логіки керування рухом каретки, обробку сигналів з датчиків позиції та аварійних ситуацій, а також інтеграцію з існуючими системами автоматизації.

В результаті проведеної роботи була створена функціональна та ефективна конвеєрна система, яка відповідає вимогам сучасного виробництва. Розроблене програмне забезпечення забезпечує надійне та оптимальне керування системою, що дозволяє підтримувати стабільну та продуктивну роботу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЬ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. 29 с.

2. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» Навчальний посібник / І. Ш. Невлюдов, В.А. Андрусевич, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків : Видавництво Іванченка І. С., 2022. 151 с.

3. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В. Токарєва,А.І. Бронніков. Харків: ХНУРЕ, 2022. 66 с.

4. Івасишин, В.І., Мовчан, Р.В. Програмування контролерів Siemens. Київ: Наукова думка, 2019. 250 с.

5. Siemens. SIMATIC S7-1200 Programmable controller [Електронний ресурс]. URL: https://support.industry.siemens.com/cs/documentation (дата звернення: 15 травня 2024).

6. Промислова робототехніка: навчальний посібник / За ред. Л.І. Цвіркуна. Вінниця: ВНТУ, 2017. 224 с.

7. Siemens. TIA Portal V15 – Getting Started [Електронний ресурс]. URL: https://support.industry.siemens.com/tia-portal-v15 (дата звернення: 20 травня 2024).

8. Brunton, S.L., Kutz, J.N. Data-Driven Science and Engineering. Cambridge University Press, 2019. 600 с.

9 Робототехніка: підручник / Під заг. Ред. О.М. Цимбала. Харків: ХНУРЕ, 2019. 248 с.

10. Wengle, M., Dalm, K., & Sahuji, R. (2023). Implementation of a Prototype

Production Line based on concept of Industrial Digitalization in an existing Learning Factory environment. Available at SSRN 4456952.

11. "Охорона праці: навчальний посібник" О. В. Пожарова. – Одеса, 2022. 86 с.