МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Ігор Невлюдов, Сергій Новоселов,
Оксана Сичова, Сергій Теслюк

Промислові мережі та компоненти АСУТП

Навчальний посібник

ВИДАВНИЦТВО ХАРКІВ 2025 ІВАНЧЕНКА І. С.

УДК 681.51
Н40

Рекомендовано Вченою радою
Харківського національного університету радіоелектроніки
(протокол №7/10 від 29.05.2025 р.)

Невлюдов І. Ш.

Н40 Промислові мережі та компоненти АСУТП : Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова, С. І. Теслюк. – Харків: Видавництво Іванченка І. С., 2025 . – 242 с.

ISBN 978-617-8332-83-9

DOI: 10.30837/978-617-8332-83-9

У навчальному посібнику подано теоретичні основи цифровізації виробництва, принципи роботи віртуальних приладів і цифрових двійників, а також засоби промислової комунікації на базі протоколу Modbus. Окрему увагу приділено практичному застосуванню середовища програмування CODESYS для створення, налаштування та управління автоматизованими системами керування технологічними процесами.

Навчальний посібник призначено для підготовки здобувачів освіти денної та заочної форм навчання спеціальності G7 Автоматизація, комп'ютерно-інтегровані технології та робототехніка. Може бути корисний аспірантам і фахівцям у промисловості, робота яких пов'язана з розробкою та організацією виробництв в галузі автоматизації та приладобудування.

УДК 681.51

Рецензенти:

– Филипенко О.І., д-р техн. наук, професор, декан факультету Автоматики і комп'ютеризованих технологій ХНУРЕ;

– Карпов Г.В., канд. техн. наук, Начальник технічного відділу ТОВ «Харківський завод спеціальних машин»;

– Артюх Р. В., канд. техн. наук, доцент, директор ДП «Південний державний проектно-конструкторський інститут авіаційної промисловості»

Industrial networks and components of process control systems: Textbook / I. Nevliudov, S. Novoselov, O. Sychova, S. Tesliuk. – Kharkiv: Ivanchenko I. Publishing House, 2025. – 242 p.

ISBN 978-617-8332-83-9
DOI: 10.30837/978-617-8332-83-9

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АСУ – автоматизована система управління;

АСУТП – автоматизована система управління технологічними процесами;

АЦП – аналого-цифровий перетворювач;

ВМ – виконавчий механізм;

ВП – виконавчий пристрій;

ВС – виробнича система;

КІТАР – кафедра комп’ютерно-інтегрованих технологій, автоматизації та робототехніки;

КУД – кінцеве устаткування даних;

ОПД – обладнання передачі даних;

ПЗ – програмне забезпечення;

ПЗП – постійний запам'ятовуючий пристрій;

ПК – персональний комп’ютер;

ПЛК – програмований логічний контролер;

ППЗП – перепрограмований постійний запам'ятовуючий пристрій;

ЦАП – цифро-аналоговий перетворювач;

ЧМн – частотна маніпуляція;

ЧС – частотний сигнал;

ADU (Application Data Unit) – елемент даних додатка;

API (Application Programming Interface) – інтерфейс прикладного програмування;

ASI (Actuator Sensor Interface) – інтерфейс датчиків і виконавчих пристроїв;

CAN (Control Area Network) – локальна мережа контролерів;

CD (Carrier Detect) – виявлення несучої;

CPS (Cyber-Physical System) – кіберфізичні системи;

CTS (Clear to Send) – вхідний сигнал для керування потоком;

DCE (Data Communications Equipment) – обладнання передачі даних;

DSR (Data Set Ready) – вхідний сигнал встановлення зв’язку;

DT (Digital Twin) – цифровий двійник;

DTA (Digital Twin Aggregate) – агреговані цифрові двійники;

DTE (Data Terminal Equipment) – кінцеве устаткування даних;

DTI (Digital Twin Instance) – цифровий двійник-екземпляр;

DTP (Digital Twin Prototypes) – цифровий двійник-прототип;

DTR (Data Terminal Ready) – вихідний сигнал встановлення зв’язку;

ETX (End of Tetx) – кінець тексту;

FSK (Frequency Shift Keying) – частотна маніпуляція;

GND (Signal ground) – загальна опорна напруга;

HART (Highway Addressable Remote Transducer) – віддалений адресний перетворювач з високою пропускною спроможністю;

Hex (Hexadecimal) – позначення шістнадцяткової системи числення;

HMI (Human Machine Interface) – людино-машинний інтерфейс;

IDE CODESYS – інтегроване середовище розробки додатків для програмованих контролерів;

IIoT (Industrial Internet of Things) – промисловий інтернет речей;

IPC (Inter-Process Communication) – набір засобів обміну повідомленнями між процесами;

ISO (International Organization for Standardization) – міжнародна організація зі стандартизації;

LiDAR (Light Detection and Ranging) – технологія отримання та обробки інформації про віддалені об'єкти за допомогою активних оптичних систем;

LLC (Logical Link Control) – підрівень керування логічним зв'язком;

MAC (Media Access Control) – управління доступом до посередників;

ModBus – відкритий комунікаційний протокол;

MTU (Maximum Transfer Unit) – максимального розміру блок корисного навантаження пакету (в байтах), який можна передати на канальному рівні;

NP (Named Pipes) – іменовані канали;

OSI (Open Systems Interconnection) – базова еталонна модель взаємодії відкритих систем;

PCM (Pulse Code Modulation) – імпульсно-кодова модуляція;

PDU (Protocol Data Unit) – елемент даних протоколу;

PLM (Product Lifecycle Management) – управління життєвим циклом продукції;

Profibus (PROcess FIeld BUS) – відкритий стандарт польової шини;

RD (Receive Data) – отримання даних;

RFID (Radio Frequency IDentification) – радіочастотна ідентифікація;

RI (Ring Indicator) – вхідний сигнал;

RTS (Request to Send) – запит на передачу;

RTU (Remote Terminal Unit) – віддалений термінальний пристрій;

RXD (Receive Data) – приймання даних;

SCADA (Supervisory Control And Data Acquisition) – програмний засіб диспетчерського управління і збіру даних;

SOH (Start Of Header) – початок заголовка;

STX (Start Of Text) – початок тексту;

TD (Transmit Data) – передача даних;

UART (Universal Asynchronous Receiver-Transmitter) – універсальний асинхронний приймач / передавач;

USB (Universal Serial Bus) – універсальна послідовна шина.

ВСТУП

На сучасному етапі цифрової трансформації виробництва, розвитку промислової автоматизації, виробничого управління та втілення концепції Індустрії 4.0/5.0, одним з найбільш важливих завдань постає впровадження новітніх технологій створення автоматизованих систем управління технологічними процесами (АСУТП) у виробництво.

В даному навчальному посібнику особлива увага приділяється питанням використання цифрових двійників (Digital Twins, DT) реальних пристроїв в процесі виконання завдань автоматизації технологічних процесів для підвищення продуктивності інтелектуального виробництва. Створення цифрових двійників дозволяє відтворити віртуальну модель реального об'єкта, процесу чи системи, яка відповідає фізичним характеристикам та поведінці реального аналога, працюючи в реальному часі за допомогою даних з сенсорів, IoT-пристроїв та інших джерел. Використання цифрових двійників дозволяє забезпечувати більш ефективне управління виробничими процесами та отримувати прогнози на основі наявних даних, тестувати об'єкти та системи в цифровому середовищі, для подальшого втілення оптимізованих рішень в реальних умовах.

Інтелектуалізація сучасних виробництв потребує інтеграцію швидкісних промислових мереж із традиційними протоколами RS-232/422/485 та Ethernet. Це дозволяє засобам автоматизації, таким як ПЛК, операторські панелі, вимірювальні прилади тощо, використовувати існуючи методи передачі даних для спільної роботи з високошвидкісними технологіями телекомунікації, надаючи можливості вже існуючим виробництвам для поступового переходу до використання хмарних технологій.

Матеріал, який подано в даному навчальному посібнику, призначено допомогти читачам отримати достатній рівень компетентності в галузі промислової автоматизації, а саме в вирішені питань застосування промислових мереж та компонентів для побудови сучасних АСУТП.

Авторами розглянуто базові технології передачі даних, сучасні інтерфейси та протоколи організації обміну інформацією в виробничому середовищі. В якості інструментів для практичної реалізації отриманих знань застосовуються: інтегроване середовище програмування CODESYS; віртуальні

лабораторні макети, розроблені співробітниками кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) для надання можливості студентам працювати в дистанційному та індивідуальному режимах навчання; емулятори протоколу Modbus в режимах головного та підлеглого пристроїв.

Велика увага приділяється питанню створення технологічних програм для ПЛК в основі яких закладається можливість управління виконавчими пристроями за допомогою промислового протоколу Modbus та отримання даних від датчиків периферійних пристроїв. Одним з сучасних інструментів програмування ПЛК, згідно з міжнародним промисловим стандартом IEC 61131-3, є середовище CODESYS. Дане програмне середовище дає змогу створювати код програми та графічний інтерфейс панелі оператора для роботи з різними типами ПЛК. Завдяки своїй гнучкості і простоті використання, середовище CODEDYS дозволяє реалізовувати рішення для різних завдань у сфері Індустрії 4.0/5.0. В навчальному посібнику описується технологія організації зв'язку між інтегрованим середовищем CODEDYS та цифровими двійниками лабораторних макетів для виконання практичних та лабораторних занять здобувачами в закладах освіти.

Навчальний посібник містить сім основних розділів.

Перший розділ присвячено опису віртуальних приладів та цифрових двійників технологічного обладнання АСУТП. Наведено основні характеристики Індустрії 4.0/5.0, розглянуто виробничі процеси та типи контролю станів технологічних процесів, а також описані віртуальні прилади, які можна використовувати в навчальному процесі: «Світлова колона», «Штампувальний автомат», «Конвеєр та технологічна лінія».

У другому розділі розглядаються питання організації взаємодії між пристроями в промислових мережах, еталонна модель взаємодії відкритих систем OSI, методи передачі даних та характеристики протоколів у промислових мережах.

Третій розділ більш детально надає інформацію щодо використання промислових інтерфейсів для управління засобами автоматизації, зокрема аналізуються такі інтерфейси, як RS-232, RS-422, RS-482 та RS-485. Описано основні принципи поєднання двох або більше пристроїв в мережу та методи організації передачі сигналів.

Четвертий розділ демонструє організацію обміну даними за протоколом Modbus для управління промисловими об’єктами. Подано приклади роботи з регістрами та функціями для роботи з дискретними та аналоговими входами та виходами ПЛК.

П’ятий розділ присвячений використанню протоколу Modbus для керування засобами автоматизації, управління віртуальними приладами з поданими прикладами для різних варіантів реалізації автоматизованих систем. Надано опис апаратного та програмного забезпечення для реалізації взаємодії між ПК та засобами автоматизації із застосуванням протоколу Modbus та інтерфейсу RS-485.

Шостий розділ дозволяє ознайомитись з середовищем програмування промислових контролерів CODESYS, починаючи зі створення та налаштування проєкту, прикладами роботи в онлайн-режимі та в режимі емуляції, закінчуючи питаннями створенням повноцінного графічного інтерфейсу користувача панелі оператора. Описані методики можна використовувати для керування, як локальними так і віддаленими засобами автоматизації, а також їх віртуальним аналогами.

В сьомому розділі розглянуто приклади управління віртуальними пристроями в середовищі CODESYS з використанням протоколу Modbus, для передачі даних, на прикладах віддаленого управління макетами «Світлова колона» і «Конвеєр та технологічна лінія».

Навчальний посібник призначено для підготовки здобувачів освіти за спеціальністю G7 Автоматизація, комп’ютерно-інтегровані технології та робототехніка, інженерів та технічних спеціалістів у сфері автоматизації та управління виробничими процесами. Також буде корисний аспірантам і фахівцям промислової галузі, робота яких пов’язана з розробкою й організацією виробництв галузі автоматизації та приладобудування.

1 ВІРТУАЛЬНІ ПРИЛАДИ ТА ЦИФРОВІ ДВІЙНИКИ ТЕХНОЛОГІЧНОГО ОБЛАДНАННЯ АСУТП

1.1 Головні характеристики Індустрії 4.0/5.0

Четверта промислова революція або перехід до Індустрії 4.0 / Індустрії 5.0 передбачає максимально широке застосування інформаційних технологій на виробництві [1].

Індустрія 4.0 – це більше, ніж бачення майбутнього. Інтелектуальна мережа являє собою величезні можливості для промисловості. Гнучке виробництво може допомогти оптимально використовувати можливості промислового обладнання.

Перша промислова революція почалася з винаходом парового двигуна в кінці XVIII століття і переходу від ручної праці до механічного виробництва (рис. 1.1).

Рисунок 1.1 – Стадії розвитку промисловості

Друга промислова революція настала приблизно через 100 років із застосуванням конвеєрного виробництва з електроприводом. З першої третини XX-го століття воно зробило можливим економічно ефективне серійне виробництво.

Електронні системи управління, інформаційні технології, електроніка, роботи і розширення використання датчиків роблять можливим подальшу автоматизацію виробничих, складальних і логістичних процесів та перехід до третьої промислової революції.

Важливо розрізняти терміни «Четверта промислова революція» і «Індустрія 4.0» (Industry 4.0) [1]. Перший визначає впровадження нових технологій 4.0 і їх вплив на всю економіку і соціальну сферу – розумні міста, будинки, сільське господарство, енергетика, інфраструктурні об'єкти, фінанси, державне управління, охорона здоров'я, освіта тощо. Індустрія 4.0 належить, насамперед, до сфери виробництва матеріальних продуктів (рис. 1.2).

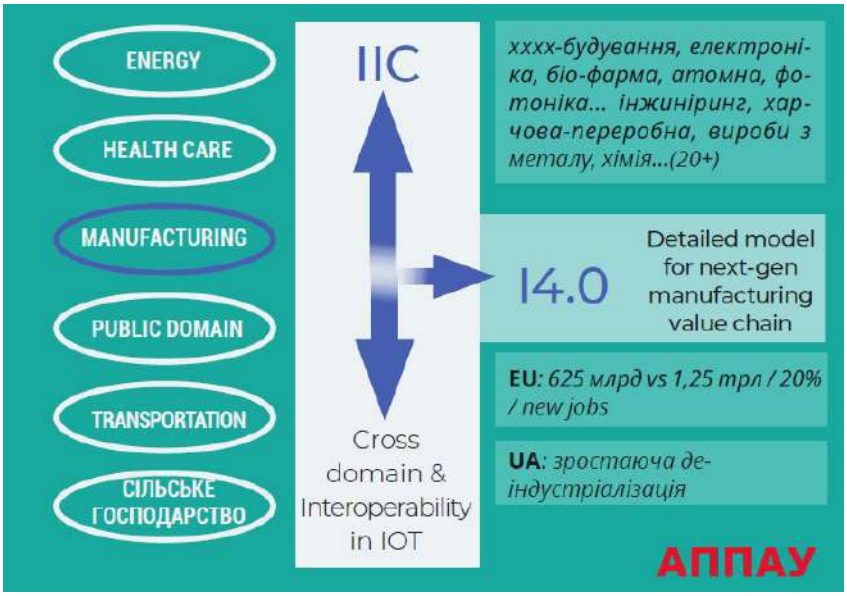


Рисунок 1.2 – Різниця між термінами «Четверта промислова революція» і «Індустрія 4.0»

Водночас, невірно ізолювати Індустрію 4.0 від інших сфер економіки. Delloite вказує, що Розумні Фабрики є дотичними до багатьох сфер, пов'язаних з промисловими виробництвами, і утворюють цілісну технологічну екосистему (рис. 1.3).

Більшість експертів в області світової Індустрії 4.0 єдині в розумінні трьох загальних характеристик (рис. 1.4). Цей фреймворк також показує, що 4.0 певною мірою є еволюцією, продовженням 3.0.

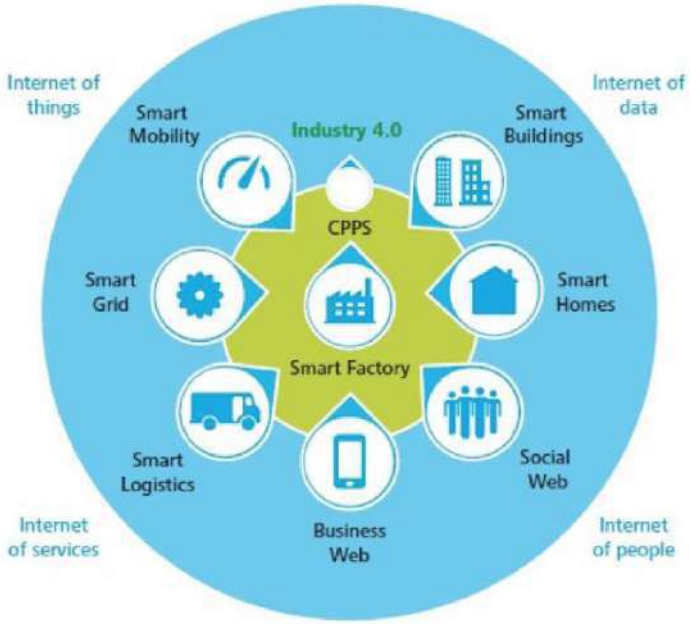


Рисунок 1.3 – Цілісна технологічна екосистема

Інші важливі характеристики, які не наведені на рис. 1.4 і які в результаті роблять фабрики і заводи «розумними», це:

– інтероперабельність: кіберфізичні системи дозволяють людям і розумному обладнанню (фабрикам) ефективніше поєднуватися один з одним;

– віртуалізація: в 4.0 можна створювати віртуальні копії розумних фізичних об'єктів (масштабованих від окремих пристроїв або машин до цілих заводів) і, відповідно, запускати різні механізми симуляцій, моделювання, а також оцінки реального стану;

– децентралізація: на відміну від високоцентралізованих підходів, у 3.0 і 4.0 кожна кіберфізична підсистема може робити власні рішення і взаємодіяти з іншими найбільш оптимальним способом;

– реальний час: всі дані та їх аналітику можливо отримувати в реальному часі;

– орієнтація на сервіси: кількість різних сервісів, як із взаємодії пристроїв і систем між собою, так і з взаємодії з людьми та учасниками екосистеми, зростає в рази;

– модульність: гнучка адаптація розумних фабрик до зовнішніх змін так само зростає, оскільки можна легко змінювати або розширювати окремі модулі систем керування.

У той час, коли зазначені характеристики є абсолютно новими, зв'язок 3.0 з 4.0, наведений на рис. 1.4, є вкрай важливим для цілого ряду індустрій. Він

стверджує, що не можна перестрибувати через рівень 3.0, якого до цих пір на 100% немає в більшості промислових галузей України.

Велика частина впровадження технології 4.0 – особливо це стосується великих даних і штучного інтелекту – базується на тому, що ці дані вже оцифровані на польовому рівні. Тобто на підприємствах вже налагоджений облік і встановлені датчики.

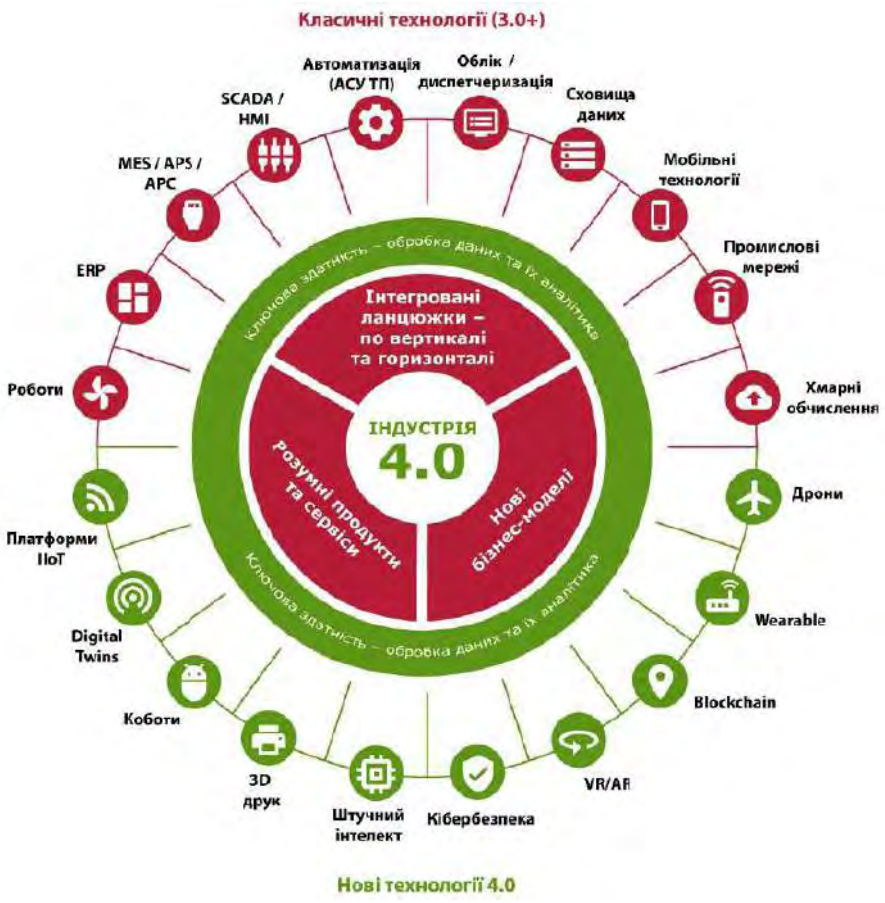


Рисунок 1.4 – Головні характеристики і технології 4.0

Індустрія 5.0 – це наступний після Індустрії 4.0 етап розвитку розумних виробництв і нова фаза індустріалізації, де фокус зміщується з аспектів цифрових технологій на чинники сталого розвитку, циркулярного виробництва та стратегічного урядування [2].

Основною тенденцією Індустрії 5.0 є запровадження спільного робочого середовища людини та робота, а також створення розумного суспільства.

Індустрія 5.0 базується не лише на технологіях, а і на таких принципах, як людиноцентричність, збереження навколишнього середовища та соціальна користь. Ця переорієнтація ґрунтується на уявленні про те, що технологія може

бути адаптована для заохочення цінностей, і що технологічні інновації можуть базуватися на етичних цілях, а не навпаки.

Впровадження штучного інтелекту, 3D-друку, віртуальної реальності та адаптивного виробництва, які є частиною нинішньої промислової революції, реалізують концепцію персоналізації продуктів та послуг відповідно до вимог клієнтів чи підприємств, дозволяючи галузі дотримуватися правильного виробничого процесу.

Ця нова революція з впровадженням нових та передових технологій допомагає виробничим процесам та суспільству виробляти більш спеціалізовані, персоналізовані, індивідуальні продукти та послуги з кращими умовами праці, використовуючи інтерактивну взаємодію між людиною та машиною.

Серед ключових переваг Індустрії 5.0 можна назвати такі:

– співпраця між людьми та машинами. На відміну від Індустрії 4.0, яка фокусувалася на повній автоматизації, Індустрія 5.0 підкреслює важливість співпраці між людьми та робототехнікою. Роботизовані системи стають більш адаптивними та гнучкими, здатними працювати поруч з людьми без ризику для безпеки;

– сталість та відповідальність. Індустрія 5.0 наголошує на необхідності відповідального використання ресурсів і сталого розвитку. Це означає не тільки мінімізацію впливу на довкілля, але й розробку продуктів з можливістю повторного використання або перероблювання.

– персоналізація виробництва. Завдяки технологіям, які дозволяють швидке переналаштування виробничих ліній, Індустрія 5.0 дозволяє масове виробництво персоналізованих товарів. Це не тільки збільшує задоволеність споживачів, але й відкриває нові можливості для нішового маркетингу.

– використання Big Data та AI. Дані та штучний інтелект продовжують грати ключову роль в Індустрії 5.0, дозволяючи не тільки оптимізувати виробничі процеси, але й передбачати тенденції ринку та налаштовувати виробництво під потреби споживачів.

Попри численні переваги, Індустрія 5.0 також стикається з викликами. Найголовнішим з них є потреба в розвитку навичок серед робочої сили, адже співпраця з розумними машинами вимагає нових знань та вмінь. Крім того, питання конфіденційності та безпеки даних залишаються актуальними через висхідне використання цифрових технологій.

1.2 Комунікаційні технології та цифровізація на інтелектуальному заводі

Комунікація на виробництві в рамках технологій 4.0 та 5.0 займає передове місце. В інтелектуальному виробництві обладнання, контролери і датчики взаємодіють як один з одним, так і безпосередньо з Ethernet або у хмарі. Закрита система стає відкритою. Але змінюється не тільки кількість інформації, що обробляється безпосередньо на місці. Підвищується і якість до абсолютно нового рівня [3].

Інформація про стан виробничого обладнання та пов'язане з ним прогнозування про можливі простої виробництва за допомогою інноваційних систем зворотного зв'язку подає тільки один із прикладів. Це стало можливим завдяки швидкому збільшенню обчислювальної потужності, яку також можна вже використовувати децентралізовано в периферії, на краю мережі або в основах виробництва.

Результатом цього стає більш гнучке й динамічне виробництво, яке в будь-який час може індивідуально і швидко реагувати на вимоги клієнтів.

Поширена досі форма комунікації між датчиками і блоками управління (ПЛК), і розташованими вище рівнями управління технологічними процесами, виробництвом і підприємством, є закритою системою (рис. 1.5). При цьому дані передаються від периферійних пристроїв, тобто датчиків і виконавчих механізмів, у програмований логічний контролер (ПЛК).

Рисунок 1.5 – Рівні комунікації на інтелектуальному заводі

Децентралізована обчислювальна потужність в ідеології Індустрії 4.0 переробляє дані в інформацію безпосередньо в датчику.

Рішення приймаються децентралізовано. Релевантна для технологічного процесу, виробництва і підприємства інформація направляється безпосередньо в Ethernet і хмарний сервіс.

На рис. 1.6 подана схема децентралізованого управління виробництвом.

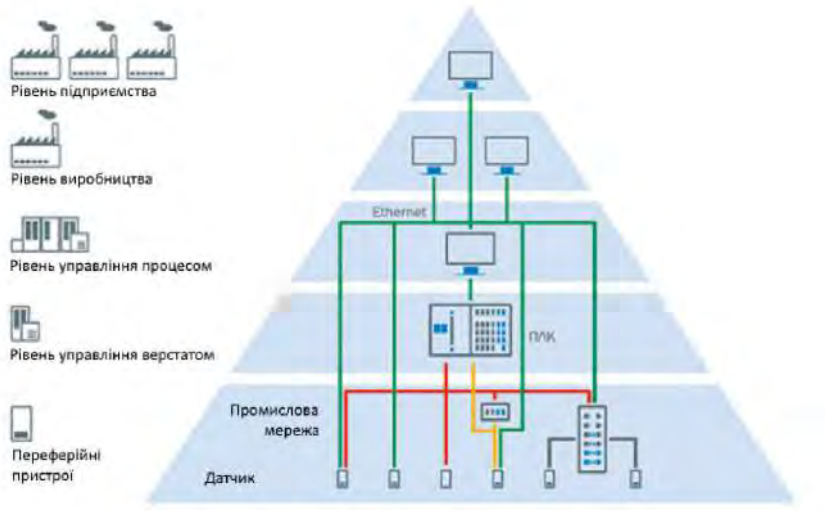


Рисунок 1.6 – Схема децентралізованого управління виробництвом

Подальший розвиток цифрового виробництва передбачає перенесення обчислень у хмарний сервіс, як подано на рис. 1.7.

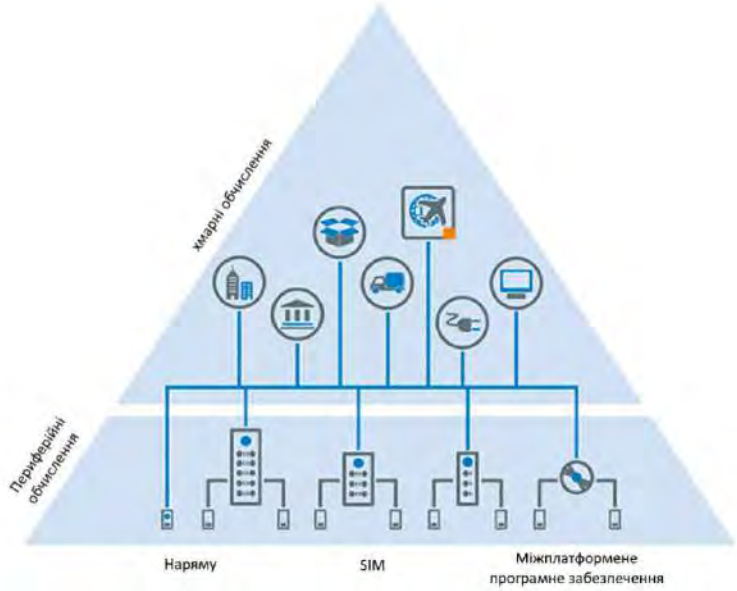


Рисунок 1.7 – Мережна інформація на цифровому виробництві

Таким чином, хмарний сервіс стає все більш важливим для управління загальними процесами. Але основна обчислювальна потужність все більше переміщується на периферію. Датчики готують зібрані дані для передачі інформації, яка потім обробляється в Ethernet або у хмарному сервісі для подальшого процесу.

1.3 Виробничі процеси та контроль стану технологічного процесу

1.3.1 Прозорі виробничі та логістичні процеси

Позитивні ефекти об'єднання в мережу в рамках Індустрії 4.0/5.0 описуються як послідовне прозоре виробництво на всьому виробничому процесі [4].

При успішному об'єднанні в мережу всіх компонентів такий вид прозорого виробництва забезпечує огляд усіх виробничих і логістичних процесів за усім ланцюжком поставок, аж до оформлення замовлень і доставки продукції замовникам (рис. 1.8). Це зменшує споживання матеріалів і ресурсів. Додатково цілісно оптимізуються виробничі і постачальні мережі.

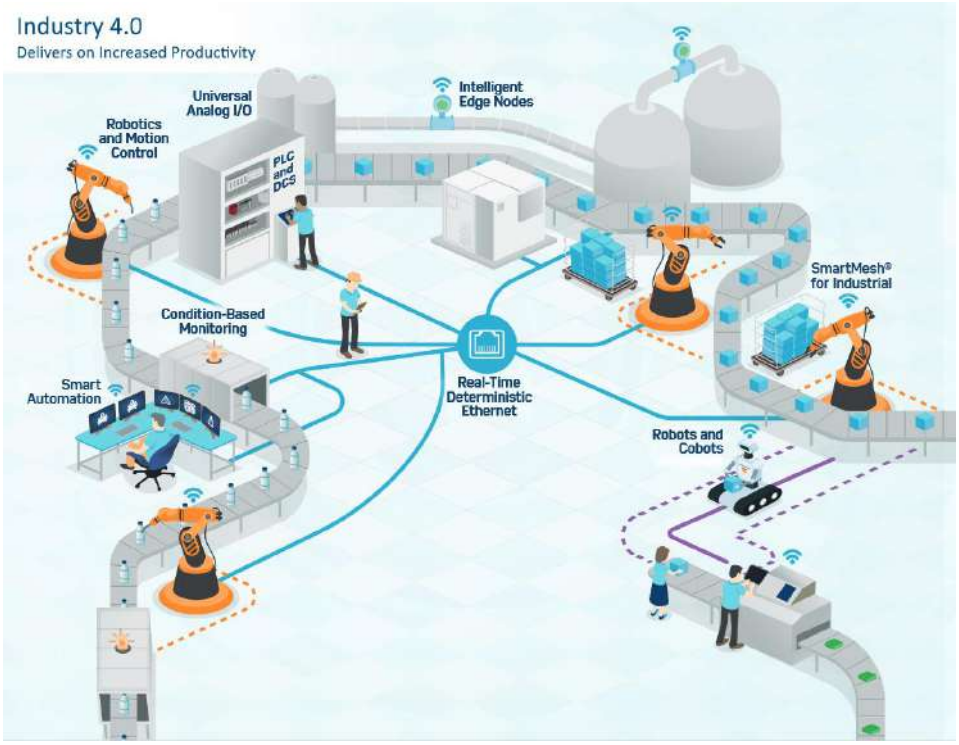


Рисунок 1.8 – Прозорі виробничі та логістичні процеси

Інтелектуальні рішення з відстеження та контролю за проходженням генерують дані та інформацію, які в мережному технологічному ланцюжку забезпечують повне виявлення, ідентифікацію та відстеження продукту і матеріалу.

Технічні можливості щодо реалізації рішень з відстеження та контролю за проходженням різноманітні. Вибір відповідної технології ідентифікації для найкращої ефективності зчитування і системної інтеграції варіюється в залежності від вимог.

Для використання в Індустрії 4.0/5.0 рішень на розумному заводі використовуються, перш за все, RFID і програмовані камери.

Датчики по всьому виробничому ланцюжку на основі носіїв даних безпосередньо розпізнають, які кроки зі збірки необхідно ініціювати, і забезпечують постійну прозорість аж до відвантаження.

Сьогодні інтелектуальні датчики означають не тільки збір даних про реальність, а й відповідну підготовку інформації вже в датчику.

Так, наприклад, за допомогою гнучкого формату виведення за допомогою налаштування і зв'язування логічних умов виведення даних можна точно привести у відповідність до вимог.

На цьому фоні кожна технологія матиме своє повноваження в майбутньому: RFID, наприклад, дозволяє зчитування і запис а, отже, багаторазове використання носіїв даних, крім того, немає необхідності в прямому «візуальному контакті».

З іншого боку, зчитувачі коду на основі камери також зчитують 2D-коди і звичайний текст. Збережені зображення можуть бути заархівовані та проаналізовані.

1.3.2 Динамічне і гнучке виробництво

Прогресивна автоматизація сприяє гнучкому виробництву і дуже невеликим обсягам партій. Побажання клієнтів стають головними. Розмір партії в одну штуку більше не є дорогою складністю, а повсякденним успішним бізнесом.

Невеликі обсяги партій та індивідуалізовані продукти масового виробництва є девізами Індустрії 4.0/5.0. Для її виправдання машина або устаткування повинні вміти обходитися зі змінною подачею продукту і бути адаптованими до різних форматів. Тільки тоді можна гнучко й ефективно

виробляти товари відповідно до побажань замовника, навіть розміром партії в одну штуку, і в адаптації до коливань попиту.

Інтелектуальні датчики забезпечують нову якість гнучкості. Вони створюють дані з виробництва в режимі реального часу. Датчики підтримують і спрощують обробку даних за рахунок того, що результати вимірювань обробляються за допомогою інтелектуальних функцій одразу на місці і передаються у вигляді підготовленої інформації, що містить тільки корисні дані.

Зі збільшенням міри автоматизації устаткування також ростуть і завдання окремих компонентів: у різних галузях вже використовуються, наприклад, фотоелектричні датчики з гнучким налаштуванням і діагностичними функціями.

Наприклад, індуктивні датчики наближення з IO-Link вирішують складні завдання безпосередньо в самому датчику. Датчики контрасту, датчики рівня та електронні реле тиску обмінюються даними налаштувань параметрів за допомогою інтегрованих інтерфейсів IO-Link.

Вимірювальні високоавтоматизовані світлові завіси знижують витрати на проводку у виробничих умовах і забезпечують доступ до діагностики та переналаштування формату.

Енкодери з EtherNet/IP™ мають як активний веб-сервер, так і функціональні блоки для інтеграції в польову шину.

Компактні датчики 2D-LiDAR (також лазерні 2D-сканери) надійно виявляють об'єкти під час контролю території.

Стандарт зв'язку з датчиками IO-link дає можливість легкої заміни кінцевих пристроїв [5]. Головний пристрій завантажує параметри у встановлений, як заміна, пристрій, що значно спрощує технічне обслуговування – не потрібне додаткове налаштування. Серед доступних різновидів датчиків IO-Link: виконавчі датчики, вимірювальні датчики, фотоелектричні датчики, RFID-датчики, лазерні датчики відстані, датчики кольору. Стандарт IO-Link регулюється відкритим консорціумом виробників.

Комунікації IO-Link на фізичному рівні реалізуються за допомогою звичайного провідникового кабелю і стандартних конекторів, таких як M12, M8 і M5. Система IO-Link складається з пристроїв IO-Link, включаючи датчики і приводи, а також провідного пристрою. Оскільки IO-Link має архітектуру типу «точка-точка», тільки один пристрій може бути під'єднаний до кожного порту головного пристрою.

Головні пристрої IO-Link можуть бути різними, включаючи карту в стійці ПЛК, або окремий пристрій з інтерфейсами до стандартних польових шин, таких, як Profibus або PROFINET.

Протокол IO-Link підтримує різні типи даних – циклічні (процесні), ациклічні, сервісні, події. Пристрій IO-Link посилає дані тільки після запиту від головного пристрою IO-Link. Ациклічні дані і події безпосередньо запитуються головним пристроєм, а циклічні дані відправляються після телеграми IDLE від головного пристрою.

Об'єднання в мережу всіх задіяних пристроїв і безпечний, децентралізований обмін даними призводить до широкого спектру варіантів застосування. Їх можна запропонувати як через хмару, так і через програмовані логічні контролери на рівні машин і систем.

Покращені обчислювальні можливості також змінюють візуальні здібності рішень на основі камери для забезпечення якості та управління виробництвом на основі датчиків.

Забезпечення якості є необхідною умовою для стійкого бізнесу і стабільного доходу. Воно включає в себе як управління матеріалами, так і функціональний контроль, контроль за машиною і виробництвом. Це дозволяє знизити рівень складських запасів і скоротити час обробки.

Сенсорні рішення для контролю за технологічним процесом і забезпечення якості підвищують гнучкість завдяки автономному коригуванню зі зміною якості і зміною продукту. Вони забезпечують ресурсоефективність, скорочення виробничого браку і високу пропускну здатність.

1.3.3 Взаємодія людини і робота на мережному заводі

У розумному підприємстві (Smart Factory) люди і роботи зближуються ще дужче [6]. У розумінні сучасного розподілу праці датчики підтримують роботів у їх роботі – даючи їм очі для виконання завдань у промислових умовах. Більш сильна взаємодія між людиною і машиною вимагає абсолютно надійних і гнучких рішень для забезпечення безпеки. Кооперація і співіснування мають стати справжньою співпрацею.

Розумне підприємство робить ставку на тісну взаємодію роботів з людьми. У цих взаємодіючих сценаріях сила, швидкість, траєкторії руху робота і заготовки становлять небезпеку для робітника. Ці небезпеки мають бути обмежені застосуванням спеціальних запобіжних заходів.

Уже сьогодні датчики безпеки дозволяють реалізувати тонку адаптацію під поточний автоматизований процес.

Інтелектуальні алгоритми дозволяють, наприклад, перехід від техніки безпеки з цифровою технологією перемикання до безперервної машинної реакції залежно від руху людини.

Отже, наближення робочого більше не призводить до загального відключення машини, а скоріше до розумного зниження робочої швидкості або коригування напрямків руху.

«Safe Motion» нині є найкращою технологією. Вона в будь-який час забезпечує безпеку людей і, незважаючи на це, немає необхідності переривати виробництво. Внаслідок цього значно скорочується час простою і кількість помилкових відключень, часи циклів стають коротшими, а ефективність та експлуатаційна готовність машин і систем збільшуються.

Якщо людина і машина співпрацюють щільно й надійно, функціональна безпека в сучасних виробничих системах є важливим кроком до більшої гнучкості.

На шляху до повноцінного співробітництва – люди і роботи ділять між собою один і той самий робочий простір і працюють у ньому водночас – існують також рішення для співіснування або кооперації.

За одночасний захист великої кількості небезпечних об'єктів відповідає, наприклад, програмований контролер безпеки з супровідним програмним забезпеченням, також у поєднанні з безпечним каскадом датчиків.

Небезпечні зони, входи і небезпечні об'єкти абсолютно надійно захищає нове покоління лазерних сканерів безпеки. Високопродуктивні світлові завіси безпеки підходять як компактні альтернативи вибіркового відключення без додаткових датчиків, а також для забезпечення високодоступного захисту небезпечних об'єктів і зон.

1.3.4 Інтелектуальні датчики як невід'ємна складова частина Індустрії 4.0/5.0

Цифровізація та об'єднання обладнання в мережу – головна риса четвертої та п’ятої промислової революції. Нові технології роблять можливим злиття фізичного і віртуального світу в області виробництва і логістики в кіберфізичні системи (CPS). З 2011 року ця розробка об'єднується під терміном «Індустрія 4.0». Машини можуть автономно обмінюватися одна з одною даними

і, таким чином, оптимізувати перебіг процесів. При цьому Індустрія 4.0/5.0 чітко посилається на створення мереж у промисловому секторі.

Сенсорна техніка створює умови для прозорих процесів в Індустрії 4.0/5.0. При цьому датчик утворює основу всіх наступних застосувань.

На відміну від класичних, немережних датчиків, датчики Індустрії 4.0/5.0 надають більше, ніж просто дані вимірювань. Інтегрована, децентралізована обчислювальна потужність і гнучка програмованість є ключовими характеристиками, які роблять виробництво більш гнучким, динамічним і ефективним.

Технологічний прогрес завжди є передумовою для змін у промисловості. Основою для динамічних, оптимізованих у реальному часі і самоорганізованих промислових процесів є отримання інформації та її подальша обробка. У них датчики виконують роль постачальників даних для інтелектуального виробництва. Сенсорна техніка є обов'язковою умовою для успішної реалізації Індустрії 4.0/5.0 (рис. 1.9).

Мережний завод є обов'язковою умовою для Індустрії 4.0/5.0. Кожен сенсор, кожна машина і всі залучені люди в будь-який час можуть спілкуватися між собою. Але цей обмін інформацією не закінчується у заводських воріт. Взаємодія між периферією і хмарним сервісом дозволяє управляти виробництвом і даними навіть ззовні.

Таким чином, це інтенсивне співробітництво між технологією і людиною робить процес більш прозорим, продуктивним і прибутковим.

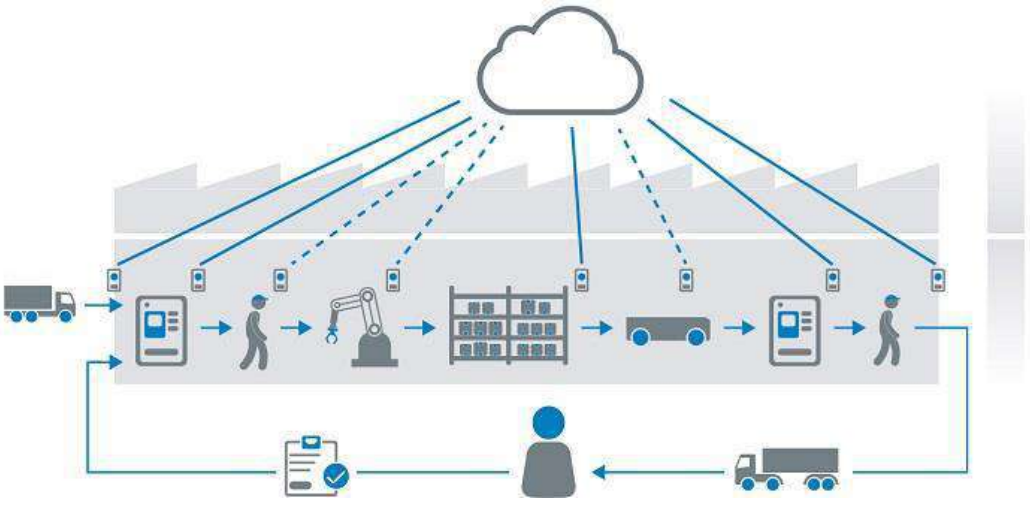


Рисунок 1.9 – Інтелектуальні датчики на мережному заводі

1.4 Визначення цифрового двійника

Цифровий двійник (Digital Twin) – програмний аналог фізичного пристрою, що моделює внутрішні процеси, технічні характеристики і поведінку реального об’єкта в умовах впливу перешкод та навколишнього середовища. Важливою особливістю цифрового двійника є те, що для завдання на нього вхідних впливів використовується інформація з датчиків реального пристрою, який працює паралельно. Робота можлива як в онлайн, так і в офлайн режимах. Далі можливе проведення порівняння інформації віртуальних датчиків цифрового двійника з датчиками реального пристрою, виявлення аномалій та причин їх виникнення [7].

Встановлення датчиків на реальний пристрій здійснюється у процесі впровадження на підприємстві технологій промислового Інтернету речей (IIoT).

Без створення цифрових двійників виробів неможливе впровадження сучасної технології PLM (Product Lifecycle Management, управління життєвим циклом виробу). IIoT і PLM – невід’ємні атрибути «розумної фабрики» (Smart Factory). Її характерна риса – формування і використання цифрової моделі матеріальних потоків, тобто, цифрового двійника вже не окремого виробу, а виробничої системи.

Якщо для традиційної промисловості набуття необхідних характеристик виробу ведеться через численні натурні випробування, то завдання Індустрії 4.0/5.0 – проводити багаторазові випробування за допомогою цифрового двійника, а натурні випробування проходити з першого разу.

Цифровий двійник виробу містить:

– геометричну і структурну модель об’єкта;

– набір розрахункових даних деталей, вузлів і виробів загалом;

– математичні моделі, які описують всі фізичні процеси, що відбуваються у виробі;

– інформацію про технологічні процеси виготовлення та збирання окремих елементів і виробу в цілому;

– систему керування життєвим циклом виробу.

Цифровий двійник об’єкта – це засіб доступу до інформації про життєвий цикл та єдиний інтерфейс до неї. Цифрові двійники можуть бути створені для будь-якої сутності, яка цікавить підприємство. Цифровий двійник складається із даних, цифрових моделей та сервісних інтерфейсів (рис. 1.10).

Рисунок 1.10 – Структура цифрового двійника об’єкту

Цифровий двійник застосовується на всіх стадіях життєвого циклу виробу, включаючи проєктування, виробництво, експлуатацію та утилізацію.

На етапі ескізного проєктування створюються варіанти комп’ютерної моделі виробу, яка розробляється для оцінки і вибору можливих технічних рішень.

На етапі технічного проєктування вибраний на попередньому етапі варіант доопрацьовується та уточняється з використанням моделей елементів. Отримана в результаті модель виробу дозволяє врахувати та оптимізувати взаємодію всіх елементів з урахуванням режимів роботи та впливів навколишнього середовища, її вже можна називати цифровим двійником виробу, який розробляється.

На етапі виготовлення розроблена модель допомагає визначити необхідні допуски при виготовленні для досягнення необхідних характеристик і забезпечення безвідмовної роботи виробу протягом усього терміну служби, а також дозволяє швидко виявляти причини несправностей у процесі тестування.

На етапі експлуатації модель цифрового двійника може бути доопрацьована та використана для реалізації зворотного зв’язку з метою внесення коректив у розробку й виготовлення виробів, діагностику та прогнозування несправностей, підвищення ефективності роботи, для виявлення нових запитів споживачів.

Цифрові двійники можна класифікувати за наступним принципом:

– цифрові двійники-прототипи (Digital Twin Prototypes, DTP);

– цифрові двійники-екземпляр (Digital Twin Instance, DTI);

– агреговані цифрові двійники (Digital Twin Aggregate, DTA).

DTP містить інформацію, необхідну для опису та створення фізичних версій екземплярів виробу. Ця інформація містить геометричну та структурну моделі, технічні вимоги та умови, вартісну модель, розрахункову (проєктну) і технологічну моделі виробу. DTP можна вважати умовно-постійною віртуальною моделлю виробу.

DTI виробу описують конкретний фізичний екземпляр виробу, з яким двійник залишається пов'язаним протягом усього терміну служби. Двійники цього типу створюються на базі DTP і додатково мають виробничу й експлуатаційну моделі, які містять історію виготовлення виробу, застосування матеріалів і комплектуючих, а також статистику відмов, ремонтів, заміни вузлів і агрегатів тощо. Таким чином, DTI виробу змінюється відповідно до змін фізичного екземпляра під час його експлуатації.

DTA виробу визначається як інформаційна система управління фізичними екземплярами сімейства виробів, яка має доступ до всіх його цифрових двійників.

Класифікація двійників виробничої системи:

– цифровий двійник усієї виробничої системи (ВС);

– цифровий двійник виробничої лінії;

– цифровий двійник конкретного активу у виробничій лінії.

1.5 Опис віртуальних приладів

На кафедрі комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) Харківського національного університету радіоелектроніки [8, 9] в лабораторії «Промислової автоматизації та мехатроніки ім. С. В. Денисова» (https://tapr.nure.ua/dijalnist-kafedri/materialno-tehnichne-zabezapechennja/nashi-laboratorii) в навчальний процес впроваджено декілька спеціалізованих навчальних стендів, в яких промислові програмовані логічні контролери та засоби автоматизації поєднані з навчальними макетами. Це і стенд фірми ОВЕН «Автоматизація управління вентиляцією на виробничому підприємстві», Навчально-методичні лабораторні комплекси від компанії «CAMOZZI Automation» – «Пневматика», «Пневмоприводи та мехатроніка».

Також в лабораторії власноруч викладачами та студентами створюються комплекти навчального обладнання і виконавчих пристроїв для проведення лабораторних робіт. За допомогою таких комплектів поступово створюється

діюча модель автоматизованої ділянки збирання засобів радіоелектроніки для моделювання основних принципів роботи сучасного виробництва в рамках концепції Industry 4.0/5.0.

Вже виготовлено і впроваджено в навчальний процес такі макети: навчальний програмований контролер; модуль дискретного вводу-виводу; модуль аналогового вводу; макет штампувального автомату; макет світлової колони; модульний ПЛК в основі якого працює міні-ПК Raspberry PI.

Перевага лабораторних макетів – це можливість наочно побачити результати роботи програми для ПЛК в дії на прикладі реального обладнання [10].

Особливості застосування макетів – це їх обмежена кількість і можливість працювати з ними за розкладом протягом лабораторних занять, або після навчання в наукових студентських гуртках. Але, як показує практика, деякі студенти мають бажання продовжувати дослідження вже вдома, виконуючи індивідуальні завдання, або вивчаючі позапланові матеріали. В таких випадках постає закономірне питання – як забезпечити кожного, хто бажає, персональним лабораторним стендом? Відповідь знаходиться на поверхні: використовувати віртуальні аналоги реальних макетів.

Концепція віртуальних макетів передбачає можливість як найповніше імітувати поведінку реальних пристроїв. Це стосується і анімації переміщення рухомих частин приладу, і способів підключення до самого макету.

Деякі інтегровані середовища розробника мають вбудовані засоби візуалізації, за допомогою яких інженер може створювати графічні інтерфейси керування та відображення інформації, що отримується від датчиків. Наприклад, в IDE CODESYS такий компонент Visualisation, який дозволяє створювати різноманітні екрани візуалізації за допомогою графічних примітивів.

Існують також інструменти, які дозволяють лише писати програмний код на одній з технологічних мов програмування, або на традиційній для програмістів мові, але не мають ніякої можливості візуального відображення стану роботи обладнання.

Запровадження дистанційного навчання в останні роки внесло свої корективи в графік навчального процесу та показало важливість кожної складової в підготовці сучасних фахівців hi-tech спеціальностей. Вже на перших тижнях гостро відчувалось відсутність можливості використання спеціалізованих макетів студентами вдома при виконанні лабораторних робіт.

Тому було вирішено створити універсальну програмну платформу для віртуалізації вже існуючих лабораторних макетів і розроблення нових. В основу платформи була закладена можливість взаємодії з віртуальним пристроєм через декілька комунікаційних протоколів: Serial Protocol, Modbus TCP/IP, Inter-Process Communication.

Крім того, брались до уваги компетенції професійної підготовки майбутніх фахівців, а також сучасні тенденції розроблення засобів радіоелектроніки з використанням Arduino серед студентів та навіть школярів.

Наприклад, Serial Protocol широко використовується під час поєднання периферійних пристроїв з контролерами Arduino, а також є не менш популярним в промисловій автоматизації. Реалізація даного протоколу в віртуальному макеті дає можливість використовувати його не тільки для поєднання з середовищем розробки програмного засобу і налагодження написаної програми, а й для самостійного використання програми, як незалежного віртуального пристрою.

Протокол Modbus TCP/IP використовується в промисловості для поєднання засобів автоматизації в єдину промислову мережу. В процесі створення програмного забезпечення та для перевірки правильності функціонування з певною версією ПЛК без наявності самого пристрою в різних IDE вбудовують можливість програмної симуляції. Але, наприклад, в інтегрованому середовищі CODESYS є суттєвий недолік – це неможливість протестувати алгоритм взаємодії між декількома пристроями з використанням комунікаційних протоколів, а також відсутність прямого доступу з програми, робота якої симулюється, до об’єкту керування. Використання протоколу Modbus TCP/IP в поєднанні з CODESYS Control Win V3, дає можливість поєднати середовище CODESYS і віртуальний макет безпосередньо в режимі симуляції.

В процесі навчання основам програмування ПЛК використовується програмний засіб LDmicro. Цей проєкт динамічно розвивається, має відкритий вихідний код та перетворює звичайний мікроконтролер в «промисловий» контролер, що програмується за допомогою технологічної мови LD. Наразі цей проєкт вже поширився на такі сімейства, як AVR, PIC, STM, ESP, Arduino. Таким чином, студенти можуть використовувати наявні в них модулі на базі вказаних мікроконтролерів для самостійного збирання засобів автоматизації їх власних проєктів. Наявність відкритого коду дало можливість зробити необхідні зміни в роботі програми та додати таку функцію, як обмін даними між зовсім

незалежними програмами через іменовані канали (Named Pipes) за допомогою технології IPC (Inter-Process Communication).

Поєднавши всі ці технології вдалося створити універсальну платформу для подальшого розвитку віртуалізації лабораторних навчальних макетів. Вона складається з трьох рівнів: комунікаційного, обробки даних, візуалізації (рис. 1.11).

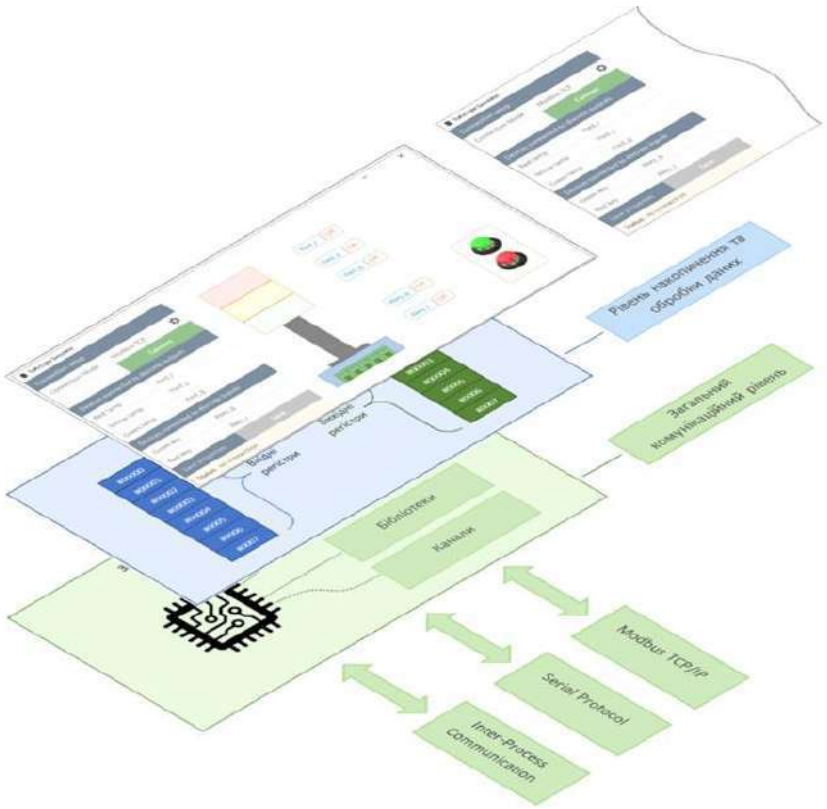


Рисунок 1.11 – Архітектура віртуальної платформи

Загальний комунікаційний рівень використовується для взаємодії зі сторонніми програмними середовищами розробки технологічних програм, або навіть з апаратними засобами.

Рівень накопичення та обробки даних використовується для обміну інформацією між нижнім і верхнім рівнями. На даному рівні виконується інтерпретація змінних, що використовуються у технологічній програмі та масивами даних, з якими працюють графічні компоненти.

Візуалізація процесу відбувається на верхньому рівні. Він створюється засобами програмування мовою C# та його зовнішній вигляд залежить від принципу дії того макету, дія якого симулюється і може бути реалізована як 3D

або 2D-анімація. За допомогою вбудованого редактора властивостей призначаються змінні та закріпляються за відповідними графічними компонентами. В подальшому, отриманні дані візуалізуються, а динамічні компоненти, наприклад, кнопки, генерують потік інформації, яка впливає на хід роботи технологічної програми. Таким чином відбувається емуляція роботи реального пристрою

Така структура дозволила розробляти різні віртуальні макети з потрібним графічним наповненням, але базуючись на загальному принципі взаємодії з засобами розробки технологічних програм, що закладено на нижніх рівнях. Поступово доповнюючи комунікаційний рівень підтримкою нових протоколів, розширюватиметься сфера застосування існуючих віртуальних макетів.

Далі наведено описи деяких прикладів віртуальних приладів, які використовуються в навчальному процесі.

1.5.1 Віртуальний прилад «Світлова колона»

Даний віртуальний прилад є цифровим двійником світлосигнальної колони для візуалізації стану промислового обладнання. На рис. 1.12 подано зовнішній вигляд інтерфейсу віртуального приладу [11].

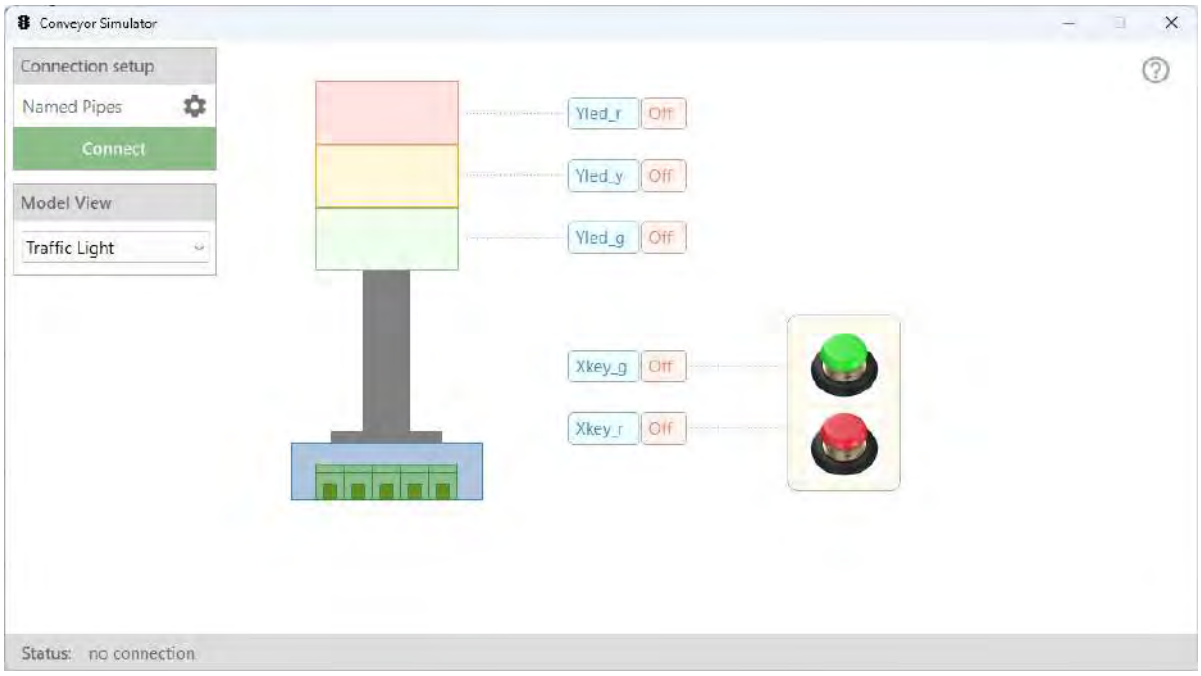


Рисунок 1.12 – Зовнішній вигляд інтерфейсу віртуального приладу «Світлова колона»

Віртуальний прилад повністю реалізує поведінку реального приладу та може працювати із зовнішньою програмою або іншим пристроєм за одним із можливих протоколів:

– Named Pipes для Inter-Process Communication;

– Modbus TCP/IP;

– Serial Protocol.

Іменовані канали (Named Pipes) забезпечують міжпроцесний зв'язок (Inter-Process Communication) між сервером каналів і одним або кількома клієнтами каналів. Вони пропонують більше функціональних можливостей, ніж анонімні канали, які забезпечують міжпроцесний зв'язок на локальному комп'ютері.

Іменовані канали підтримують повний дуплексний зв'язок через мережу та кілька екземплярів сервера, зв'язок на основі повідомлень та уособлення клієнта, що дозволяє процесам підключення використовувати власний набір дозволів на віддалених серверах. Цей тип взаємодії є основним для роботи з програмою LDmicro, що є інструментом створення технологічних програм мовою LD.

Modbus TCP/IP використовується в промисловості для поєднання засобів автоматизації в єдину промислову мережу. В віртуальному приладі наявність підтримки цього протоколу надає можливість використовувати його спільно з інтегрованим середовищем CODESYS, в якому відсутній прямий доступ до об'єкту керування з програми, робота якої симулюється. Використання посередника SoftPLC (йде в наборі утиліт разом з CODESYS) та протоколу Modbus TCP/IP надає можливість застосовувати даний віртуальний прилад в єдиному віртуальному комплексі.

Підтримка Serial Protocol надає можливість використовувати віртуальний прилад для налагодження програм разом з реальними пристроями, наприклад із Arduino. Реалізація даного протоколу в віртуальному макеті дає можливість використовувати його не тільки для поєднання з середовищем розробки програмного засобу і налагодження написаної програми, а й для самостійного використання програми, як незалежного віртуального пристрою.

В інтерактивному інтерфейсі можна виділити:

– органи керування (зелена та червона кнопки);

– органи відображення інформації (червона, жовта та зелена лампа).

Для полегшення сприйняття принципу роботи цифрового двійника, на екранній формі органи відображення інформації поєднані в єдину конструкцію, що зовні виглядає ідентично реальному пристрою.

Кнопки керування мають візуальний ефект в процесі взаємодії з ними користувача. Наведення курсора та натискання кнопки миші призводить до того, що віртуальна кнопка теж «натискається», а відпускання повертає її в нормальне положення. Віртуальна кнопка працює в режимі нормально розімкнених контактів. Її стан можна контролювати за візуальною міткою (рис. 1.13).

Рисунок 1.13 – Візуальна мітка стану елементу керування

В інтерфейсі візуальної мітки є таки елементи:

– назва змінної;

– поточний стан змінної;

– лінія поєднання з елементом керування або відображення.

Назва змінної – це ідентифікатор, що використовується в зовнішній програмі для доступу до цифрового двійника, до потрібного органу керування або візуалізації. Літера «X» в префіксі назви змінної означає, що вона пов’язана з вхідною інформацією для ПЛК (кнопка пов’язана з зовнішнім вхідним контактом ПЛК). Літера «Y» – означає, що змінна пов’язана з вихідною інформацією, яка йде від ПЛК (лампа пов’язана з зовнішнім вихідним контактом (реле) ПЛК). Назви змінних, що вказані у віртуальних приладах неможна змінювати, тому при написанні програми потрібно чітко стежити за відповідними мітками, що пов’язані з елементами керування та відображення інформації.

Поточний стан змінної залежить від стану органу керування або від результатів роботи технологічної програми. Якщо змінна пов’язана з кнопкою, або іншим органом керування на віртуальному приладі, то її стан змінюється в незалежності від того, чи підключена до віртуального приладу програма. З натисканням кнопки можна бачити, що стан змінної змінився (рис. 1.14).

З рис. 1.14 можна бачити, що натиснута зелена кнопка ініціювала зміну стану змінної «Xkey_g» на «On», в той же час, не натиснута червона кнопка представлена значенням «Off» змінної «Xkey_r».

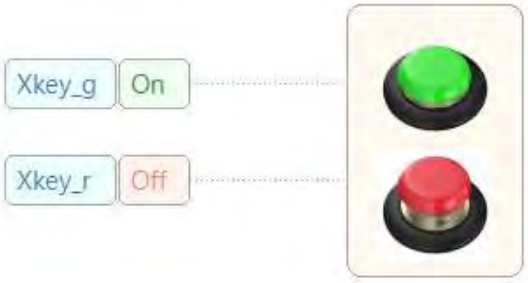


Рисунок 1.14 – Зміна стану змінної з натисканням кнопки

1.5.2 Віртуальний прилад «Штампувальний автомат»

Зовнішній вигляд інтерфейсу користувача віртуального приладу подано на рис. 1.15. Даний віртуальний прилад за функціональними можливостями відповідає реальному лабораторному макету «Штампувальний автомат», розробленому на кафедрі КІТАР.

У відповідності до єдиної концепції побудови віртуальних приладів, кожен елемент керування та відображення інформації має візуальну мітку стану (рис. 1.13).

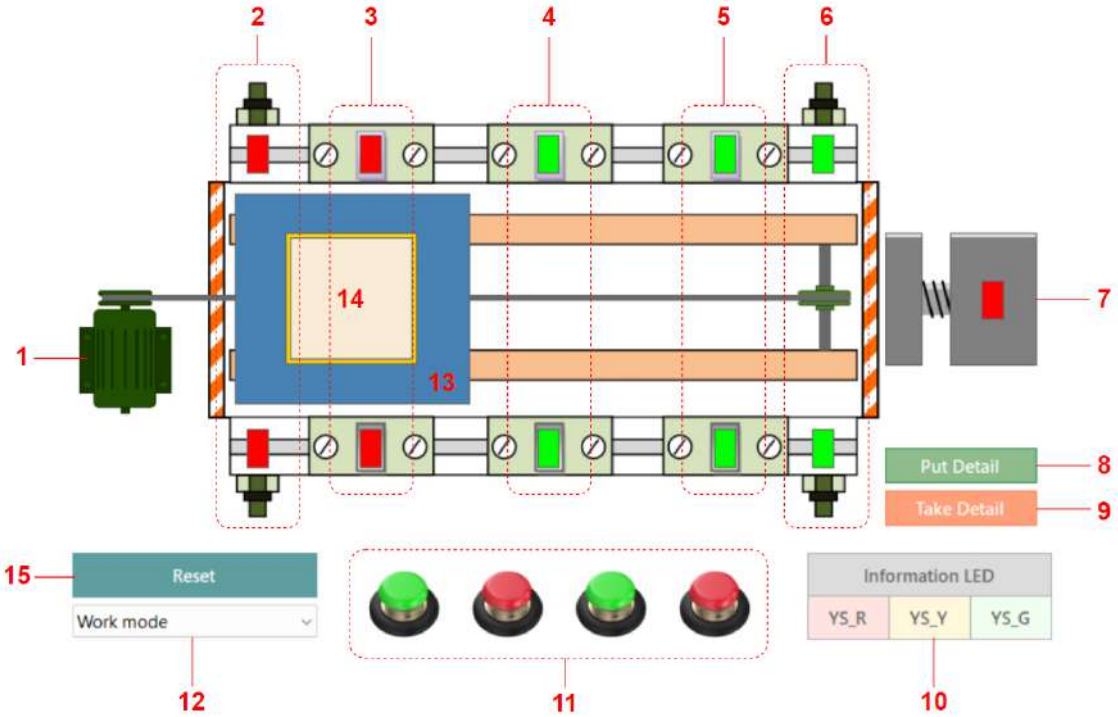


Рисунок 1.15 – Зовнішній вигляд інтерфейсу користувача віртуального приладу «Штампувальний автомат»

Теліжку 13 з деталлю 14 рухає двигун 1. Рух може відбуватись зліва на право або в зворотному напрямку. Фізичного обмеження руху не існує, це основна особливість, як фізичного макету, так і його цифрового двійника. Зупинка теліжки виконується програмним шляхом на основі даних, що отримуються з датчиків 2 та 6. Датчик 2 – це лівий кінцевий датчик, а 6 – правий.

Всі датчики в даному пристрої побудовані за принципом закритого світлового каналу та мають розгалужену в просторі структуру.

Однією з умов роботи керуючої програми є запобігання виїзду теліжки за обмежувальну лінію (рис. 1.16). Технологічна програма повинна оброблювати інформацію з кінцевих датчиків 2 та 6 і вчасно змінювати стан вихідних реле, до яких підключено двигун 1, що рухає теліжку.

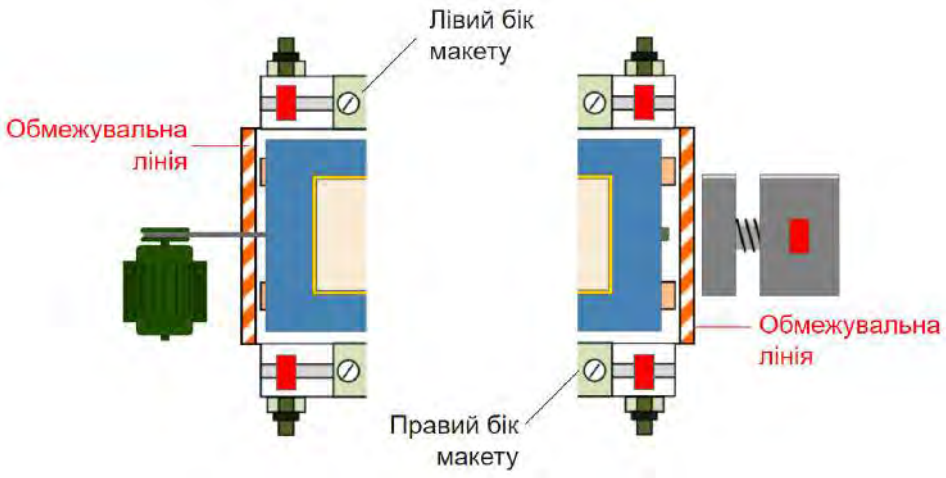


Рисунок 1.16 – Обмежувальна лінія штампувального автомата

У віртуальному приладі передбачена функція оброблення помилки керування та, в разі виїзду теліжки за обмежувальну лінію (ліву або праву), на екрані з’явиться попереджувальна інформація (рис. 1.17), а робота макету буде аварійно зупинена до натискання кнопки Reset (1.15, поз. 15).

Дана ситуація в реальній сфері застосування автоматизованого пристрою є критичною, тому в цифровому двійнику вона оброблюється на рівні основної програми управління віртуальним приладом.

Управління роботою двигуна, що приводить до руху теліжку, відбувається двома сигналами «YS_Tlg_L» та «YS_Tlg_R». Реакція двигуна на різні комбінації сигналів подана у табл. 1.1

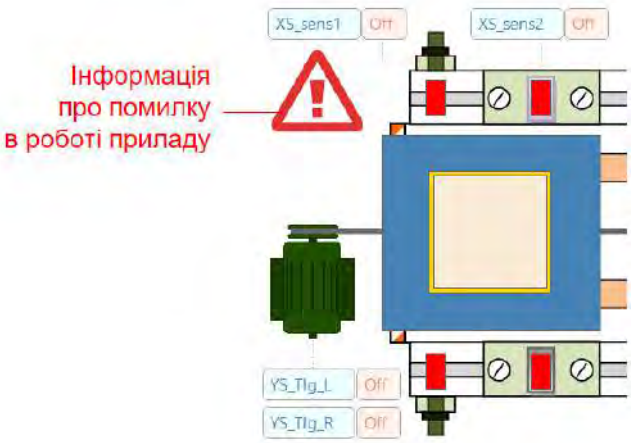


Рисунок 1.17 – Виїзд теліжки за обмежувальну лінію

Датчики 3, 4 та 5 (рис. 1.15) призначені для контролю наявності вантажу, що міститься на теліжці. Зелений колір датчика означає, що він знаходиться в стані спрацювання (між випромінювачем та приймачем немає перешкоди). Червоний колір означає, що між частинами датчика знаходиться якийсь предмет. Датчик 3 контролює зону завантаження. Датчик 4 визначає середню позицію розташування теліжки з вантажем. Датчик 5 контролює наявність деталі на теліжці в робочій зоні штампувального автомата.

Таблиця 1.1 – Режим роботи двигуна в залежності від комбінації сигналів

YS_Tlg_L	**YS_Tlg_R**	**Режим роботи двигуна**
0 (Off)	0 (Off)	Двигун не працює
1 (On)	0 (Off)	Теліжка рухається вліво
0 (Off)	1 (On)	Теліжка рухається вправо
1 (On)	1 (On)	Невизначена ситуація. Рух теліжки призупинено

Механізм штампування (рис. 1.15, поз. 7) керується вихідним сигналом «YS_Stamp». Поява логічної одиниці на цьому реле призводить включення механізму приводу штампу. Цей процес займає деякий час та не залежить від логічної одиниці на реле «YS_Stamp» після початку процесу руху робочої частини штампу. В керуючій програмі можна згенерувати короткий імпульс для вмикання даного режиму.

Робоча частина штампу повернеться в початковий стан очікування керуючого сигналу після завершення процесу штампування. Також, після

завершення даного процесу на деталі (рис. 1.15, поз. 14) з’явиться візуальна мітка, що підтверджує виконання операції (рис. 1.18).

Мітка буде очищена після ручного зняття та встановлення нової деталі на теліжку. Для керування цим процесом в віртуальному приладі передбачені дві кнопки:

– Put Detail (покласти нову деталь на теліжку);

– Take Detail (прибрати деталь з теліжки).

Прибрати деталь з теліжки можна в будь який час. Програма, що керує автоматом повинна опрацьовувати цю ситуацію та, наприклад, зупиняти рух теліжки і повертати її до зони завантаження.

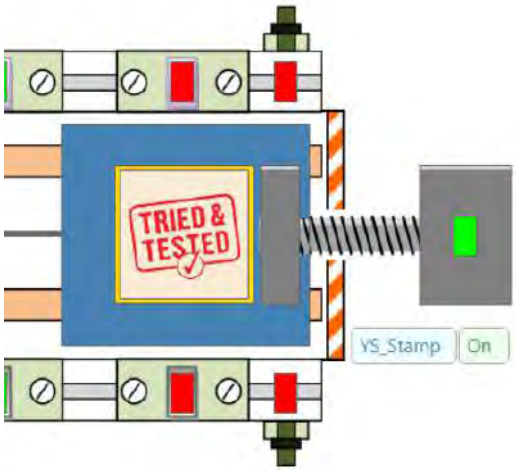


Рисунок 1.18 – Виконання процесу штампування

У віртуальному приладі додатково передбачені декілька органів керування та візуалізації для реалізації розвинутих алгоритмів керування процесом штампування. Наприклад, є інформаційне поле (рис. 1.15, поз. 10) з трьома світлодіодами, що виконує функції подібні до світлової колони (рис. 1.19).

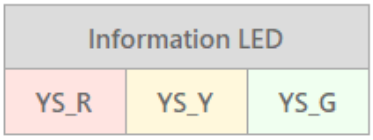


Рисунок 1.19 – Інформаційне поле з трьома світлодіодами

Дане поле має три індикатори, які керуються відповідними вхідними сигналами:

– червоний світлодіод керується сигналом «YS_R»;

– жовтий світлодіод керується сигналом «YS_Y»;

– зелений світлодіод керується сигналом «YS_G».

Також передбачено блок з чотирьох кнопок (рис. 1.15, поз. 11), який користувач може використовувати для виконання функцій, що передбачені в завданні (рис. 1.20).

Кожна кнопка з блоку прив’язана до відповідної змінної:

– кнопка 1 пов’язана зі змінною «XS_key1»;

– кнопка 2 пов’язана зі змінною «XS_key2»;

– кнопка 3 пов’язана зі змінною «XS_key3»;

– кнопка 4 пов’язана зі змінною «XS_key4».

Рисунок 1.20 – Блок кнопок користувача

Відповідні інформаційні мітки сигналізують про поточний стан кнопок. Кнопки мають нормально відкриті контакти та не мають фіксації, тобто після відпускання повертаються у вимкнений стан.

Віртуальний прилад має два режими роботи (елемент вибору на рис. 1.15, поз. 12):

– Work mode (робочий режим);

– Demo mode (демонстраційний режим).

В робочому режимі прилад реагує тільки на зовнішню програму керування, наприклад, яку завантажено в режимі симуляції в модифікованій програмі LDmicro. Принцип роботи подано на рис. 1.21.

Віртуальна програма виробляє сигнали, які пов’язані з вхідними контактами ПЛК. Технологічна програма для ПЛК опитує вхідні контакти, аналізує їх стан, та, за розробленим алгоритмом дій, виробляє вихідні сигнали на реле, що пов’язані з відповідними елементами у віртуальному приладі.

Елементи інтерфейсу, які мають назву з першою літерою «X», підключені до вхідних контактів та виробляють сигнали керування. Наприклад, такими елементами є кнопки.

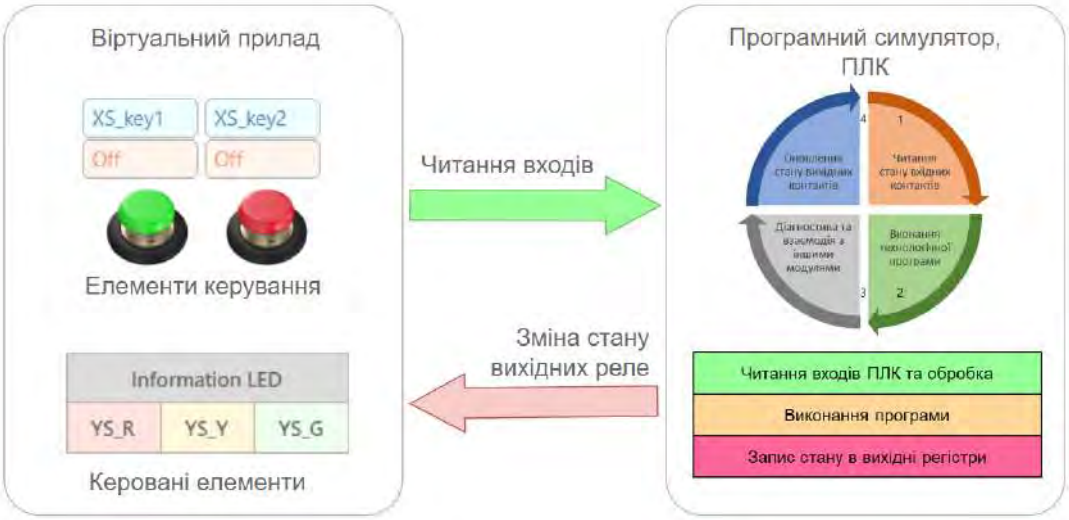


Рисунок 1.21 – Принцип взаємодії віртуального приладу і програмного симулятора ПЛК

Елементи інтерфейсу, що мають назву з першою літерою «Y», підключені до вихідних контактів та вони отримують сигнали керування від ПЛК. Такими елементами, наприклад, є двигун, штампувальний механізм, світлодіодні індикатори.

В режимі демонстрації віртуальний прилад налаштовано на виконання команд від кнопок керування без реалізації функцій контролю стану датчиків автоматизованої системи.

Після переходу в демонстраційний режим запускається двигун та теліжка рухається вправо. Для керування її рухом можна використовувати кнопки. Наприклад, натискання на кнопку 1 («XS_key1») призводить до зміни напрямку руху теліжки. Якщо вона їхала вправо, то після натискання на кнопку 1 рух зміниться на протилежний – вліво.

Кнопка 2 («XS_key2») вмикає рух механізму штампа. Якщо, в той час, коли механізм дійде до крайньої позиції, там буде знаходиться деталь, то на ній з’явиться відмітка про виконання процесу штампування.

Кнопки 3 та 4 в демонстраційному режимі не задіяні. Кнопки Put Detail (покласти нову деталь на теліжку) та Take Detail (прибрати деталь з теліжки) працюють в звичайному для них режимах.

1.5.3 Віртуальний прилад «Конвеєр та технологічна лінія»

Зовнішній вигляд інтерфейсу користувача віртуального приладу «Конвеєр та технологічна лінія» подано на рис. 1.22. Даний віртуальний прилад є цифровим двійником мехатронного модуля транспортування та розподілення фірми FESTO.

Особливістю реалізації даного приладу є поєднання конвеєрної лінії з модулем живлення деталями конвеєрної лінії в єдиний виробничій модуль.

Програмна реалізація цих двох макетів дозволяє виконувати функції:

– видачі деталей для живлення конвеєрної лінії;

– переміщення їх конвеєрною лінією;

– за потреби, розподілення потоку деталей в залежності від характеристик деталей, або за іншим принципом;

– управління режимом роботи макету за допомогою додаткового блоку кнопок;

– візуалізація поточного стану всіх елементів макету.

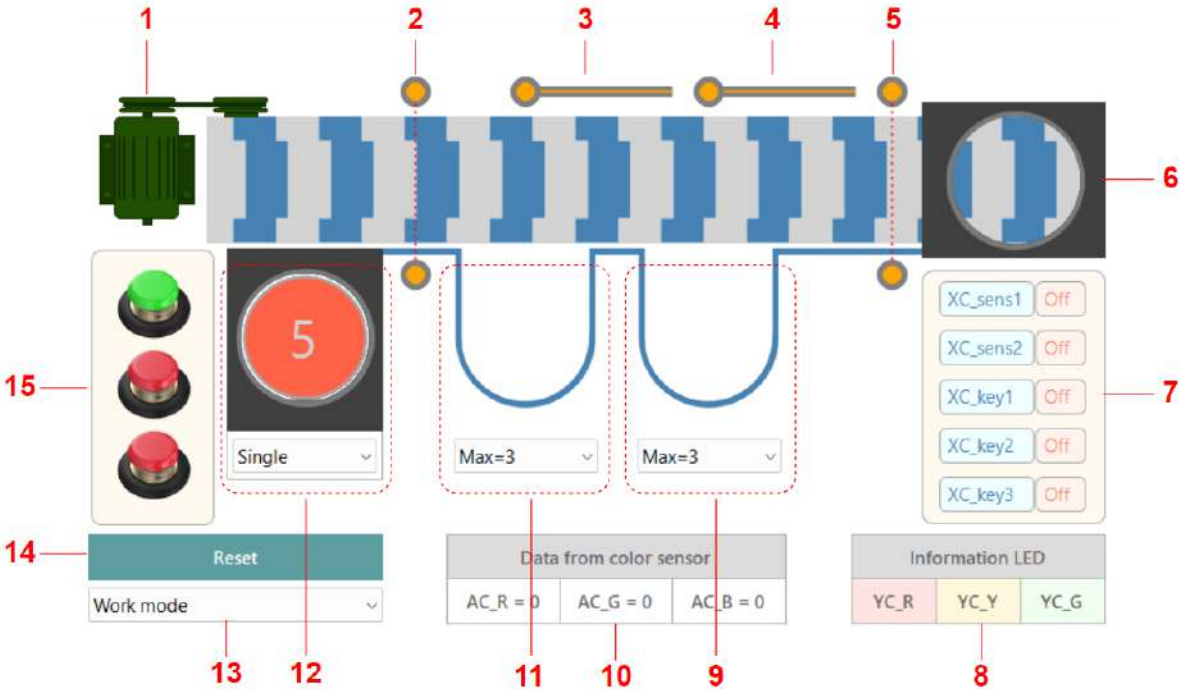


Рисунок 1.22 – Зовнішній вигляд інтерфейсу користувача віртуального приладу «Конвеєр та технологічна лінія»

Розглянемо призначення елементів керування та візуалізації стану віртуального макету. Двигун (рис. 1.22, поз. 1) рухає конвеєрну стрічку тільки в одному напрямку – зліва на право, тому для включення двигуна використовується тільки один сигнал та відповідна змінна «YC_Moto».

Віртуальний макет має три датчики:

– датчик рахунку деталей на вході конвеєра (рис. 1.22, поз. 2);

– датчик рахунку деталей на виході з конвеєра (рис. 1.22, поз. 5);

– аналоговий датчик кольору деталі (рис. 1.22, поз. 11), що конструктивно поєднаний з датчиком рахунку деталей на вході (рис. 1.22, поз. 2).

Датчики 2 та 5 дискретні. Сигнал на їх виході може бути одним з двох можливих станів: true або false (on та off).

Датчик рахунку деталей (рис. 1.22, поз. 2) видає сигнал логічної одиниці, коли вона перетинає його промінь, і утримує незмінний вихідний сигнал протягом всього часу, поки деталь не вийде за його межі. З цим датчиком пов’язана змінна «SC_sens1». Датчик 5 (рис. 1.22) працює аналогічним чином, а його стан пов’язаний зі змінною «SC_sens2».

Для визначення кольору деталей, що надходять до конвеєрної лінії використовується аналоговий датчик (рис. 1.22, поз. 11). Його розташовано поряд з датчиком 2, та кожен раз, коли деталь проходить повз нього, на виході отримуємо комбінацію із трьох сигналів R, G та B (рис. 1.23).

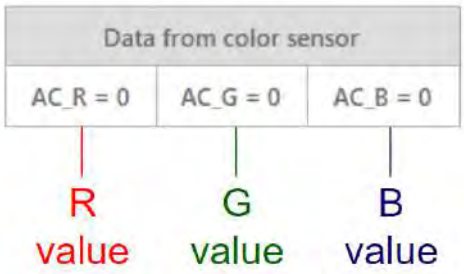


Рисунок 1.23 – Три канали датчика кольору

В датчику 11 застосовано 8-ми бітний АЦП. Кожен з трьох каналів формує на виході число в діапазоні від 0 до 255, а комбінація із трьох чисел описує колір деталі, що пройшла повз датчик.

За кожним із каналів закріплена відповідна змінна. Червоний канал пов’язано зі змінною «AC_R», зелений – «AC_G», синій – «AC_B». Перша літера

в назві змінної означає, що це дані з АЦП та за правилами програми LDmicro вони можуть зчитуватись вбудованою інструкцією ReadADC.

Рухомі заслінки 3 та 4 (рис. 1.22) призначені для розділення потоку деталей на конвеєрі. Вони можуть займати одно з двох стійких положень (рис. 1.24).

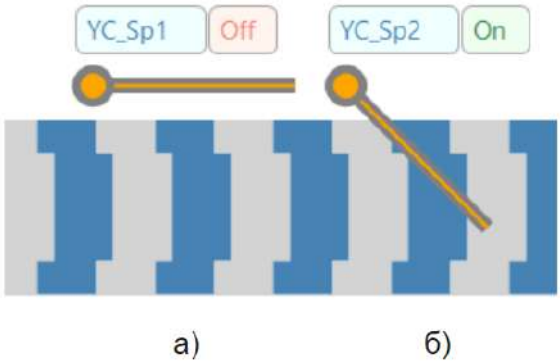


Рисунок 1.24 – Рухомі заслінки для розділення потоку деталей

В нормальному режимі, коли на привід заслінки не надходить керуючий сигнал, вона перебуває в горизонтальному положенні (рис. 1.24, а). Якщо надходить керуючий сигнал, спрацьовує привід заслінки і вона переводиться в положення, коли потік деталей перекривається під кутом 45 градусів (рис. 1.24, б). В такому її положенні деталі, що рухаються конвеєрною стрічкою, змінюють напрям руху та зміщуються в бік одного з накопичувачів 9 або 11.

Принцип дії рухомих важелів подано на рис. 1.25. З надходженням деталі в накопичувач збільшується число, яке відображає поточну кількість виробів в даній зоні.

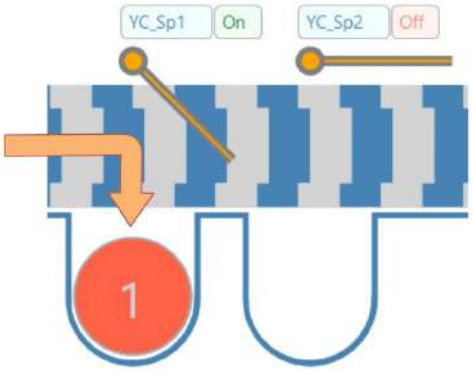


Рисунок 1.25 – Принцип дії рухомих важелів

В віртуальному макеті можна задавати максимально можливу кількість деталей в накопичувачі (рис. 1.26). В цьому разі, в керуючій програмі збоку ПЛК необхідно слідкувати за тим, щоб цей ліміт не було перевищено.

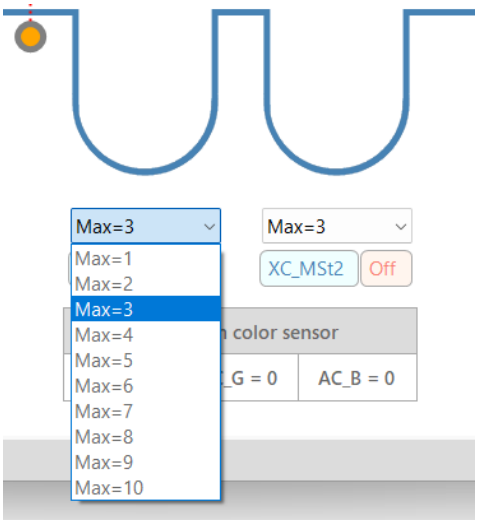


Рисунок 1.26 – Управління максимально можливою кількістю деталей в накопичувачі

У випадку, коли максимальна кількість деталей досягнута, а програма все одно перенаправляє нову деталь в накопичувач, віртуальний прилад згенерує помилку. Помилка буде показана користувачу у вигляді діалогового вікна (рис. 1.27).

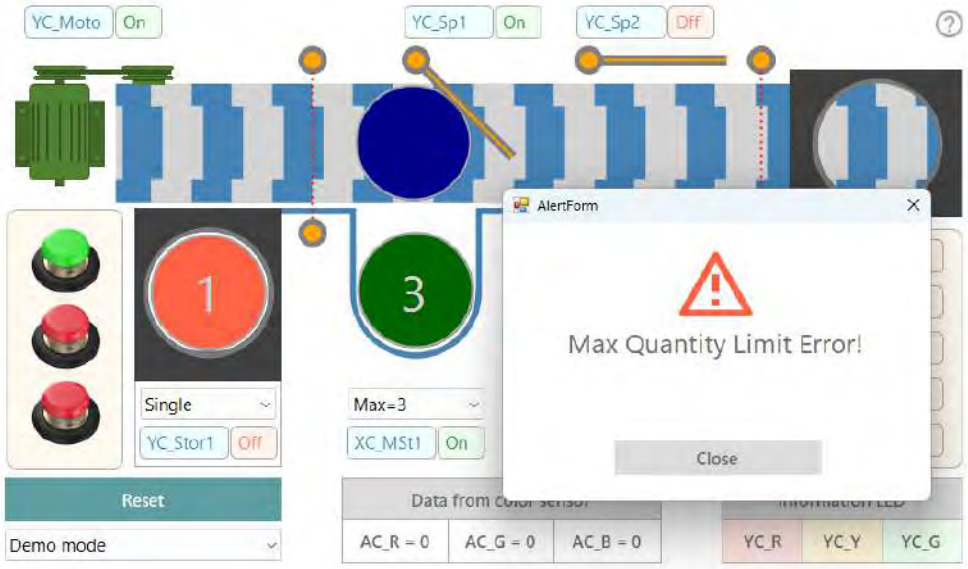


Рисунок 1.27 – Вікно, що сповіщає про помилку

З рис. 1.27 можна бачити, що для накопичувача встановлено ліміт кількості деталей, що дорівнює трьом, але програма керування направила ще одну, синю, деталь в той самий накопичувач. В даному випадку на екрані з'являється повідомлення про помилку режиму виконання «Max Quantity Limit Error!». Після натискання на кнопку «Close» потрібно перезапустити програму натиснувши на кнопку «Reset» (рис. 1.22, поз. 14).

Всі деталі, що пройшли конвеєром та не були направлені в інший потік, проходять вздовж датчика 5 та потрапляють до накопичувача 6 (рис. 1.22). Кількість деталей, що порахована датчиком 6 відображається на фоні деталі в кінцевому накопичувачі.

Магазин деталей (рис. 1.22, поз. 12) живить конвеєр виробами. Він може працювати в двох режимах (рис. 1.28):

– Single: видача деталей по одній;

– Auto: автоматична видача деталей.

Кількість деталей, що залишилась в магазині, відображається у вигляді числа, що знаходиться на фоні чергової деталі.

Рисунок 1.28 – Вибір режиму роботи магазину деталей

Видача деталей в режимі Single виконується по одній в разі надходження сигналу керування на вхід магазину. Даний сигнал пов'язано зі змінною «YC_Stor1». Сигнал логічної одиниці запускає поршень, який виштовхує деталь на конвеєр (рис. 1.29). Доти, доки механізм не завершить процес виштовхування та не повернеться у вихідне положення, він не приймає та не обробляє вхідні керуючі сигнали.

В автоматичному режимі, механізм видачі деталей працює циклічно. Після завершення видачі чергової деталі, виконується новий цикл виштовхування, доки на закінчяться всі деталі в магазині.

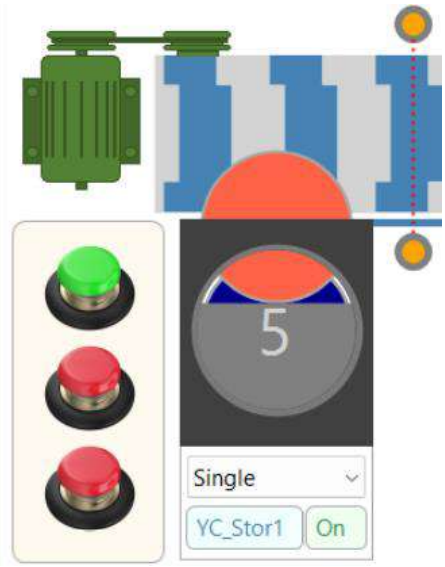


Рисунок 1.29 – Виштовхування деталі з магазину на конвеєр

Автоматичний режим працює обох режимах роботи програми – в демонстраційному режимі та в режимі симуляції.

У віртуальному макеті передбачені декілька додаткових органів керування та візуалізації для можливості впровадження різних алгоритмів керування конвеєрною лінією.

Для відображення інформації про поточний стан роботи макету можливо використовувати блок з трьох різнокольорових світлодіодів (рис. 1.22, поз. 8). Інформаційний блок має три індикатори, що керуються відповідними вхідними сигналами (рис. 1.30):

– червоний світлодіод керується сигналом «YC_R»;

– жовтий світлодіод керується сигналом «YC_Y»;

– зелений світлодіод керується сигналом «YC_G».

Рисунок 1.30 – Інформаційний блок з трьома світлодіодами

Для керування віртуальним макетом можна застосовувати три кнопки (блок кнопок на рис. 1.22, поз. 15). Поточний стан кожної кнопки можна контролювати за відповідною інформаційною міткою. Всі мітки для трьох кнопок та двох датчиків зібрані в єдиний блок (рис. 1.22, поз. 7).

Кожна кнопка з блоку прив'язана до відповідної змінної (рис. 1.31):

– кнопка 1 (зелена) пов'язана зі змінною «XC_key1»;

– кнопка 2 (червона) пов'язана зі змінною «XS_key2»;

– кнопка 3 (червона) пов'язана зі змінною «XS_key3».

При написанні програми керування необхідно мати на увазі, що кнопки мають нормально відкриті контакти та не мають фіксації – після відпускання повертаються у вимкнений стан.

В режимі демонстрації перша кнопка (зелена) запукає процес видачі нової деталі з магазину. Друга кнопка керує заслінкою (рис. 1.22, поз. 3). З натисканням цієї кнопки, заслінка переводиться в положення, що перекриває напрям руху деталі та змінює його для скидання виробу в накопичувач 11. Повторне натискання цієї кнопки повертає важіль в початкове положення і потік деталей відновлюється.

Рисунок 1.31 – Інформаційний блок для індикації стану датчиків та кнопок віртуального макету

Третя кнопка керує заслінкою 4 (рис. 1.22). З натисканням цієї кнопки, заслінка переводиться в положення, що перенаправляє деталі в накопичувач 9. Повторне натискання на кнопку повертає важіль в початкове положення.

Зміна режимів роботи відбувається з допомогою списку (рис. 1.22, поз. 13).

Щоб перевести макет в початковий стан треба натиснути кнопку «Reset» (рис. 1.22, поз. 14).

1.6 Контрольні питання

1. Які основні характеристики Індустрії 4.0 та Індустрії 5.0?
2. Що означає цифровізація виробничих процесів?
3. У чому полягає суть прозорих виробничих та логістичних процесів?
4. Як реалізується динамічне і гнучке виробництво на сучасних підприємствах?
5. Які функції виконують інтелектуальні датчики в контексті Індустрії 4.0/5.0?
6. Що таке цифровий двійник і які його основні характеристики?
7. Які переваги використання цифрових двійників у технологічному обладнанні АСУТП?
8. У чому полягає різниця між віртуальними приладами та фізичними аналогами?
9. Які функції виконує віртуальний прилад «Світлова колона»?
10. Як працює віртуальний прилад «Штампувальний автомат» і яке його призначення?
11. Яке значення має віртуальний прилад «Конвеєр та технологічна лінія» в навчальному процесі?
12. Як цифрові двійники допомагають у контролі стану технологічного процесу?
13. Який вигляд має інтерфейс користувача віртуального приладу «Світлова колона»?
14. Яким чином реалізується зв’язок між віртуальним приладом і зовнішньою програмою або іншим пристроєм?
15. Що таке візуальна мітка?
16. Як виконати підключення віртуального приладу до постачальника інформації?
17. Які основні елементи має віртуальний прилад «Штампувальний автомат»? Поясніть принцип його функціонування.
18. Поясніть принцип взаємодії віртуального приладу і програмного симулятора ПЛК.
19. Які функції може виконувати віртуальний прилад «Конвеєр та технологічна лінія»? Які датчики передбачено в даному приладі?

2 ОРГАНІЗАЦІЯ ВЗАЄМОДІЇ МІЖ ПРИСТРОЯМИ В ПРОМИСЛОВИХ МЕРЕЖАХ

2.1 Модель взаємодії відкритих систем OSI

В сучасних системах автоматизації, внаслідок постійної модернізації виробництва, все частіше трапляються завдання побудови розподілених промислових мереж із використанням гнучких протоколів передачі.

Для організації промислових мереж використовується безліч інтерфейсів і протоколів передачі даних, наприклад Modbus, Ethernet, CAN, HART, Profibus тощо. Вони необхідні для передачі даних між датчиками, контролерами і виконавчими механізмами (ВМ); калібрування датчиків; живлення датчиків та ВМ; зв'язку нижнього та верхнього рівнів АСУТП. Протоколи розробляються з урахуванням особливостей виробництва та технічних систем, забезпечуючи надійне з'єднання та високу точність передачі між різними пристроями. Поряд із надійністю роботи в жорстких умовах все більш важливими вимогами в системах АСУТП стають функціональні можливості, гнучкість у побудові, простота інтеграції та обслуговування, відповідність промисловим стандартам.

Оскільки основною функцією мережі є з'єднання між собою різного обладнання, проблема відкритості, зокрема стандартизації, для мереж набуває особливого значення.

Найбільш поширеною системою класифікації мережних протоколів є теоретична модель OSI (базова еталонна модель взаємодії відкритих систем, англ. Open Systems Interconnection Basic Reference Model). Специфікацію цієї моделі було остаточно прийнято у 1984 році Міжнародною Організацією зі Стандартизації (ISO). Відповідно до моделі OSI протоколи діляться на 7 рівнів, розташованих один над одним, за своїм призначенням – від фізичного (формування та розпізнавання електричних або інших сигналів) до прикладного (API для передачі інформації додатками).

Такий підхід забезпечив можливість вирішення завдання взаємодії систем для кожного рівня окремо, зокрема незалежними групами розробників. Зокрема, для мережної взаємодії пристроїв необхідно узгодити між собою електричні рівні сигналів, затримки та тривалості імпульсів, типи з'єднувачів, способи кодування інформації, способи забезпечення достовірності передачі, форми та

формати адресації, формати даних, способи доступу до мережі, способи буферизації даних, способи поділу їх на пакети та відновлення цілісності повідомлень та ін.

Взаємодія між рівнями може здійснюватися як вертикально, так і горизонтально (рис. 2.1). У горизонтальній взаємодії програмам потрібен загальний протокол обміну даними. У вертикальній – за допомогою інтерфейсів.

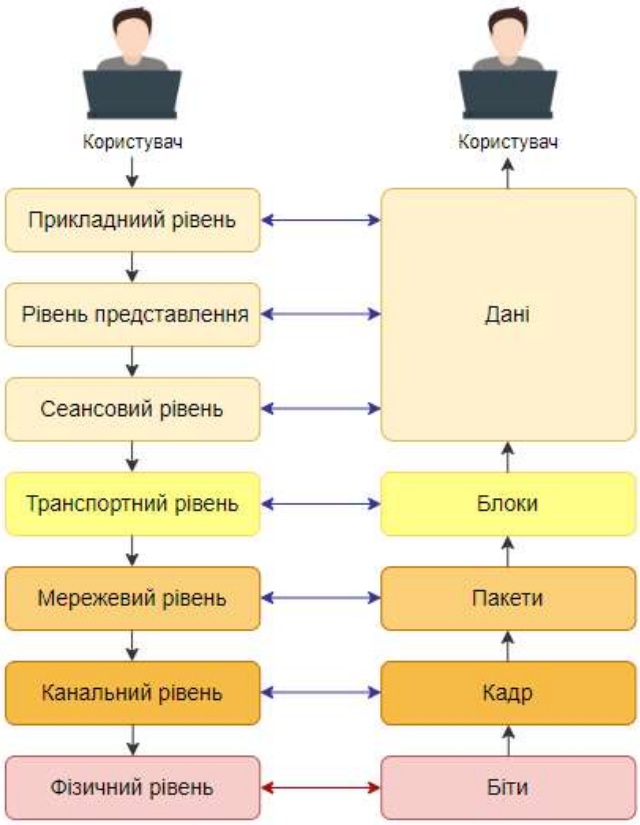


Рисунок 2.1 – Теоретична модель OSI

Прикладний рівень – рівень додатків (англ. Application layer). Забезпечує взаємодію мережі та програм користувача, що виходять за рамки моделі OSI. На цьому рівні використовуються такі протоколи: https, gopher, Telnet, DNS, SMTP, SNMP, CMIP, FTP, TFTP, SSH, IRC, AIM, NFS, NNTP, NTP, SNTP, XMPP, FTAM, APPC, X.400, X .500, AFP, LDAP, SIP, ITMS, Modbus TCP, BACnet IP, IMAP, POP3, SMB, MFTP, BitTorrent, eD2k, Profibus.

Представницький рівень (англ. Presentation layer) – рівень представлення даних. На цьому рівні може здійснюватися перетворення протоколів та стиснення / розпакування або кодування / декодування даних, а також перенаправлення запитів іншому мережному ресурсу, якщо вони не можуть бути оброблені локально. Запити програм, отримані з рівня додатків, він перетворює

на формат передачі через мережу, а отримані з мережі дані перетворює на формат, зрозумілий додаткам. До цього рівня зазвичай належать такі протоколи: https, ASN.1, XML-RPC, TDI, XDR, SNMP, FTP, Telnet, SMTP, NCP, AFP.

Сеансовий рівень (англ. Session layer) управляє створенням / завершенням сеансу зв'язку, обміном інформацією, синхронізацією завдань, визначенням права на передачу даних та підтримкою сеансу в періоди неактивності додатків. Синхронізація передачі забезпечується поміщенням потік даних контрольних точок, починаючи з яких відновлюється процес при порушенні взаємодії. Використовувані протоколи: ASP, ADSP, DLC, Named Pipes, NBT, NetBIOS, NWLink, Printer Access Protocol, Zone Information Protocol, SSL, TLS, SOCKS.

Транспортний рівень (англ. Transport layer) організує доставку даних без помилок, втрат і дублювання в тій послідовності, як вони були передані. Поділяє дані на фрагменти рівної величини, поєднуючи короткі та розбиваючи довгі (розмір фрагмента залежить від протоколу, що використовується). Використовувані протоколи: TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX, SCTP, DCCP, RTP, TFTP.

Мережний рівень (англ. Network layer) визначає шляхи передачі. Відповідає за трансляцію логічних адрес та імен у фізичні, за визначення найкоротших маршрутів, комутацію та маршрутизацію, за відстеження неполадок та заторів у мережі. Використовуються протоколи: IP, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, ARP, RARP, DHCP, BootP, SKIP, RIP.

Канальний рівень (англ. Data link layer) призначений для забезпечення взаємодії мереж фізично. Отримані з фізичного рівня дані перевіряє на помилки, якщо потрібно виправляє, пакує у кадри, перевіряє на цілісність, і відправляє на мережний рівень. Канальний рівень може взаємодіяти з одним чи кількома фізичними рівнями. Специфікація IEEE 802 розділяє цей рівень на два підрівні: MAC (Media Access Control) – регулює доступ до фізичного середовища, що розділяється, та LLC (Logical Link Control) – забезпечує обслуговування мережного рівня. Використовувані протоколи: STP, ARCnet, ATM, DTM, SLIP, SMDS, Ethernet, FDDI, Frame Relay, LocalTalk, Token ring, StarLan, L2F, L2TP, PPTP, PPP, PPPoE, Profibus.

Фізичний рівень (англ. Physical layer) призначений безпосередньо передачі потоку даних. Здійснює передачу електричних або оптичних сигналів в кабель або радіоефір і, відповідно, їх прийом і перетворення в біти даних відповідно до

методів кодування цифрових сигналів. Використовувані протоколи: RS-232, RS-422, RS-423, RS-449, RS-485, ITU-T, xDSL, ISDN, T1, E1, 10BASE-T, 10BASE2, 10BASE5, 100BASE-T, 1000 , 1000BASE-TX, 1000BASE-SX.

Перелічимо мережні протоколи, згрупувавши їх за рівнями призначення:

– фізичного рівня: RS-485, ISDN, RS-232, EIA-422;

– канального рівня: Token ring, Ethernet, FDDI, GVRP, HDLC, PPP, L2TP, PPTP, xDSL, ATM;

– мережного рівня: IPv4, IPv6, ICMP, ARP, IPX;

– транспортного рівня: SPX, TCP, DHCP (в моделі OSI), SCTP, UDP (Unreliable/User Datagram Protocol), RTCP, RUDP (Reliable User Datagram Protocol), RDP (Reliable Data Protocol);

– сеансового рівня: SSL;

– представлення даних: XML-RPC, ASN.1, XDR, TDI, FTP, SNMP, Telnet, NCP, SMTP;

– прикладного рівня: binkp, Finger, FTP, Gnutella, DNS, HTTP, Gopher, IMAP, HTTPS, XMPP, IRC, NTP, LDAP, POP3, NNTP, SSH, RDP (Remote Desktop Protocol), Telnet, SMTP, SIP, SNMP.

2.2 Методи передачі даних в промислових мережах

2.2.1 Визначення терміну «Передача даних»

Передача даних – це фізичне перенесення цифрового (бітового) потоку у вигляді сигналів від точки до точки або від точки до множини точок засобами електрозв'язку каналом зв'язку; як правило, для подальшої обробки комп'ютерами. Прикладами подібних каналів можуть бути мідні проводи, оптичне волокно, бездротові канали зв'язку.

Передача даних може бути аналоговим чи цифровим (потік двійкових сигналів), а також модульованим за допомогою аналогової модуляції, або за допомогою цифрового кодування.

Тоді як аналоговий зв'язок є передачею змінного аналогового сигналу, у цифровому зв'язку повідомлення є дискретними. Повідомлення є або послідовністю імпульсів, що означає лінійний код (у смузі пропускання), або обмежуються набором хвиль з неперервно змінними формами, використовуючи метод цифрової модуляції. Такий спосіб модуляції і відповідна йому демодуляція здійснюється модемним обладнанням.

Передані дані можуть бути цифровими повідомленнями, що йдуть від джерела даних, наприклад, з комп'ютера або від клавіатури. Це може бути й аналоговий сигнал – телефонний дзвінок або відеосигнал, оцифрований у бітовий потік, з використанням імпульсно-кодової модуляції (PCM) або більш розширені схеми кодування джерела (аналого-цифрове перетворення та стиснення даних). Кодування і декодування джерела здійснюється шифратором або кодувальним обладнанням.

2.2.2 Способи передачі даних

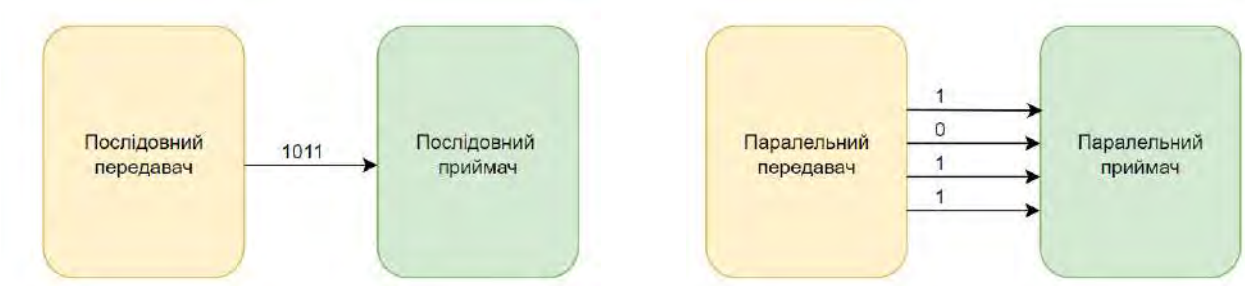
Існують два способи передачі слів інформації по лініях даних:

– послідовний, коли біти слова надсилаються по черзі;

– паралельний, коли одночасно пересилаються всі біти слова.

В процесі послідовної передачі даних надсилаються і приймаються елементи, які є символами чи іншими об'єктами даних. Цифрова послідовна передача – це послідовне відправлення бітів одним дротом, ефіром чи оптичним шляхом (рис. 2.2, а). Через те, що це вимагає менших зусиль для обробки сигналу, і при цьому менша ймовірність помилки, ніж при паралельному пересиланні, то швидкість передачі даних кожним окремим шляхом може бути більша. Цей механізм може використовуватися на більших відстанях.

Паралельною передачею в телекомунікаціях називається одночасне пересилання елементів сигналу одного символу або іншого об'єкта даних (рис. 2.2, б).



а) б)

Рисунок 2.2 – Передача комбінації 1011:
а) послідовна передача; б) паралельна передача

В цифровому зв'язку паралельною передачею називається одночасне пересилання відповідних елементів сигналу двома або більшою кількістю

шляхами. Використовуючи безліч електричних дротів, можна передавати кілька біт одночасно, що дозволяє досягти більш високих швидкостей пересилання, ніж в послідовній передачі.

Цей метод застосовується всередині комп'ютера, наприклад, у внутрішніх шинах даних, а іноді і в зовнішніх пристроях, таких, як принтери. Основною проблемою при цьому є «перекіс» в надходженні даних до приймача, тому що дроти у паралельній передачі мають трохи різні властивості, тому деякі біти можуть прибути раніше за інші, і це може пошкодити зміст повідомлення.

Додаткові засоби контроля правильності переданої інформації дозволяють виявити помилки обміну даних. Проте електричний провід в паралельній передачі даних менш надійний на великих відстанях, оскільки ймовірність порушення сигналу набагато вища.

2.2.3 Особливості реалізації асинхронної послідовної передачі даних

В послідовному обміні даними використовуються три методи передачі (рис. 2.3):

– симплексна (односпрямована) передача (телебачення, радіо);

– напівдуплексна (прийом та передача інформації здійснюються по черзі);

– дуплексна (двонаправлена) (кожна станція одночасно передає та приймає дані).

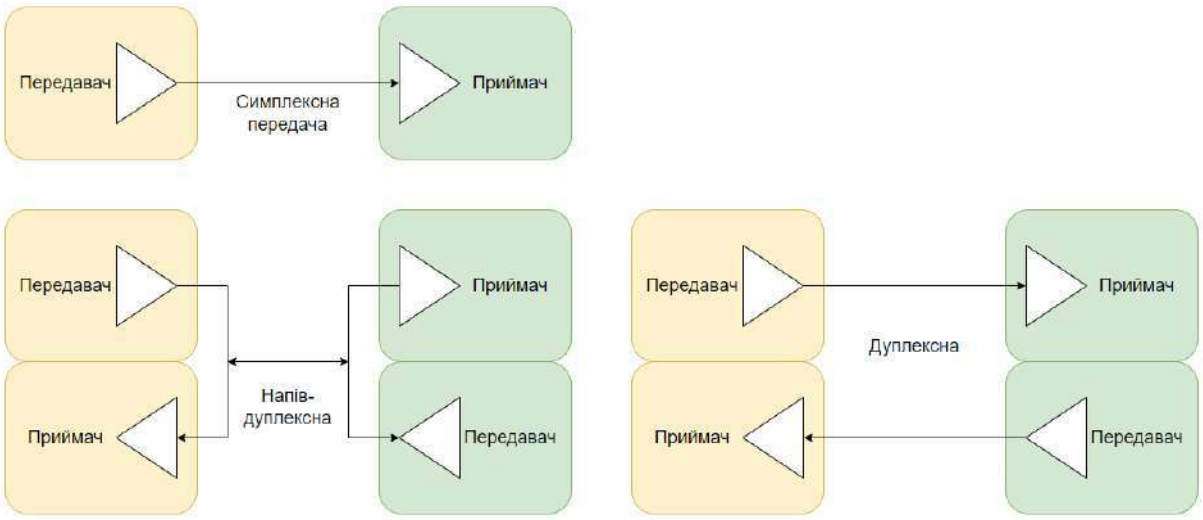


Рисунок 2.3 – Методи послідовної передачі даних

Коли для передачі даних між передавачем і приймачем доступний тільки один канал зв'язку, проблема синхронізації стає більш складною. У послідовних

системах існує два основних способи синхронізації передавача й приймача. Один з них ґрунтується на неявній синхронізації взаємодіючих пристроїв і називається асинхронним обміном даними, інший базується на явній синхронізації при обміні інформацією й, відповідно, називається синхронним обміном даними.

Суть асинхронного принципу полягає у незалежній (повністю або частково) роботі (за часом) передавача та приймача, керованої регулярною послідовністю сигналів синхронізації при реалізації функцій обміну та обробки інформації.

В асинхронних системах періоду часу, протягом якого передавач і приймач повинні бути синхронізовані, як правило, є дуже нетривалим. Наприклад, за таким методом передається один символ з кодового набору ASCII. Відповідно, термін «посимвольне пересилання» краще підходить для опису асинхронного обміну інформацією.

У таких системах, моделлю яких може служити людина, що натискає клавіші, час між передачею послідовних символів (міжсимвольний інтервал) різний, звідси й походження терміну «асинхронний». З натисканням клавіші, клавіатура генерує код, що зазвичай складається з 10 біт (стартовий біт, 7 біт даних, біт парності й стоповий біт). Щоб одержати ці 10 біт без помилок, приймач повинен отримати їх з каналу зв'язку в правильний момент.

Перш ніж розпочати послідовну передачу даних, необхідно виконати перетворення даних з паралельної форми на послідовну (рис. 2.4).

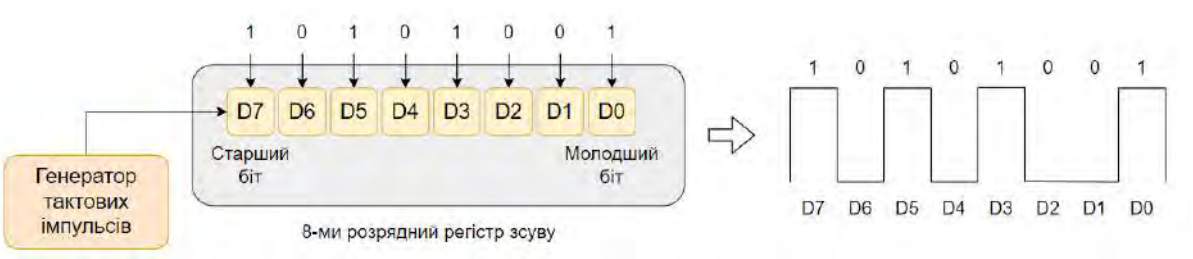


Рисунок 2.4 – Перетворення паралельних даних на послідовні

На рис. 2.5 подано принцип реалізації послідовного обміну даними між двома пристроями.

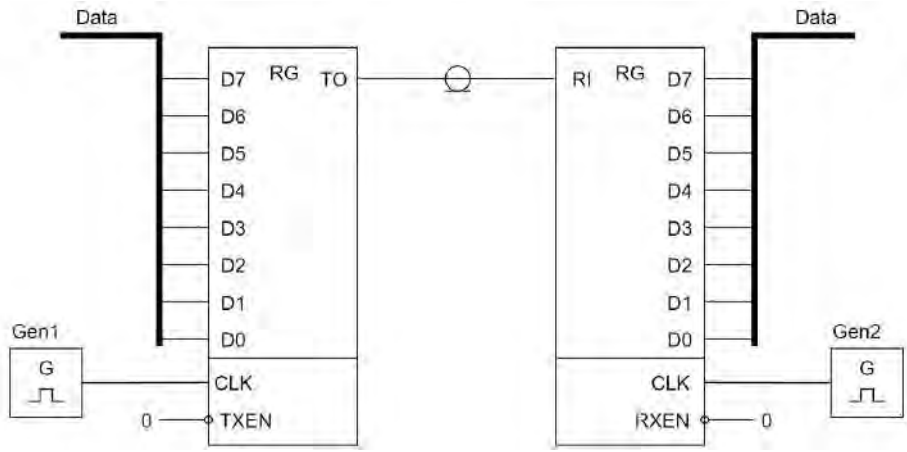


Рисунок 2.5 – Принцип послідовної передачі даних

З поданого рисунку можна бачити, що в передавачі та в приймачі необхідна наявність такого елементу, як регістр зсуву. В передавачі регістр отримує на вхід паралельний код та за сигналами тактового генератору (Gen1) почергово зсуває дані та передає їх через вихід ТО.

На боці приймача інший регістр зсуву послідовно отримує біти інформації через канал зв’язку та за імпульсами тактового генератору Gen2 почергово зсуває дані та формує з них паралельний код на виходах D0-D7.

Передавач і приймач повинні бути синхронізовані. В ідеалі приймач повинен отримувати кожний біт точно посередині інтервалу проходження цього біта, щоб прийом був максимально безпомилковим. При швидкості зв'язку 1200 біт/с кожний біт з'являється в каналі зв'язку приблизно на 830 мкс (1/1200).

Стартовий біт дозволяє вирішити цю проблему. Початок стартового біта відзначається в лінії зв'язку переходом зі стану з деякою позитивною напругою в стан відсутності напруги (тобто «земля»). Коли приймач визначає початок передачі, про що сигналізує поява стартового біта, він відраховує половину інтервалу передачі біта і здійснює першу вибірку.

Для зчитування решти дев'яти біт символу, приймач відраховує дев'ять рівних інтервалів по 830 мкс кожний, здійснюючи вибірку даних з лінії наприкінці цих інтервалів. Таким чином, передавач і приймач залишаються синхронізовані протягом пересилання символу й не мають потреби в обміні явною синхронізувальною інформацією. Звичайно, асинхронний метод обміну даними залежить від ступеня точності ходу незалежно працюючих годинників у передавачі й приймачі.

Найбільше застосування отримали старт-стопні принципи синхронізації за бітами та знаками.

Суть старт-стопного принципу управління полягає в тому, що стартовий імпульс у повідомленні запускає синхрогенератор приймача, який працює на частоті передавача, і лінія стробується відповідно до частоти місцевого синхронізатора, а стоповий імпульс у повідомленні зупиняє синхрогенератор.

Приклад формату кадра, що застосовується при асинхронній передачі даних, подано на рис. 2.6.

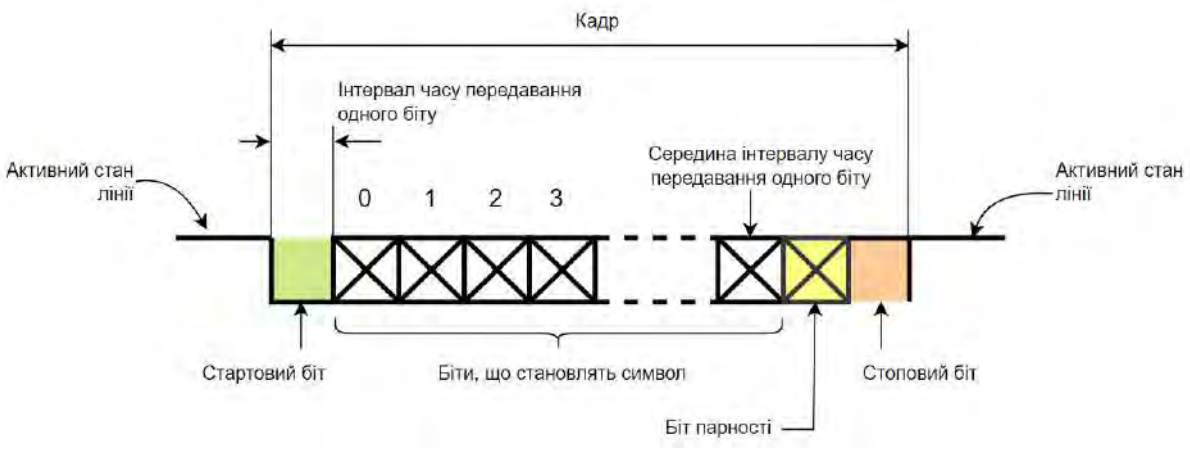


Рисунок 2.6 – Приклад формату кадру, що застосовується в асинхронній передачі даних

Передача даних здійснюється порціями (кадрами). Початок та кінець кожної порції інформації відзначаються спеціальними мітками.

Рівень логічної одиниці лінії називають маркером, рівень логічного нуля – пробілом.

За відсутності даних в лінії діє сигнал маркера. Передача кадру починається з посилки стартового біта – пробілу. Стартовий біт попереджає приймач початку передачі. Після стартового біта передаються біти даних, кількість яких у кадрі може встановлюватися від 5 до 8.

Для визначення достовірності передачі даних після останнього біта даних може бути біт паритету, який також називається бітом контролю парності (або непарності). Цей біт вибирається у кожному кадрі даних в такий спосіб, щоб загальна кількість одиниць у бітах даних і биті паритету було парним (чи непарним), тобто біт парності дорівнює 1, якщо кількість одиниць у символі непарна, і 0, якщо інакше.

Кадр закінчується стоп-бітом, який сигналізує про закінчення передачі та має рівень маркера. Може встановлюватися один, півтора або два стопові біта. Після цього в лінії може підтримуватись стан відсутності даних (рівень маркера) або починатися наступний кадр (стартовим бітом-пробілом).

Пристрій, що забезпечує перетворення даних з паралельної форми на послідовну і назад, і забезпечує асинхронний протокол обміну даними, називають універсальним асинхронним приймачем (Universal Asynchronous Receiver-Transmitter, UART).

Такий пристрій реалізується у вигляді спеціального модуля мікроконтролера або окремої мікросхеми. Крім перетворення форми подання приймач виконує важливі функції контролю та управління.

Швидкість передачі даних описаним способом прийнято вимірювати в бодах. Один бод дорівнює одному біту в секунду, і тому швидкість в бодах означає скільки біт в секунду може бути передано каналом передачі даних.

Організація асинхронного послідовного обміну даними із зовнішнім пристроєм ускладняється тим, що на передавальній та приймальній стороні послідовної лінії зв'язку використовуються налаштовані на одну частоту, але фізично різні генератори тактових імпульсів і, отже, загальна синхронізація відсутня.

Для вирішення цієї проблеми для роботи UART використовується послідовність тактових імпульсів, період проходження яких у 16 разів менше часу передачі одного біта даних (рис. 2.7). Ці тактові імпульси використовуються для синхронізації вхідного сигналу та для генерування вихідного.

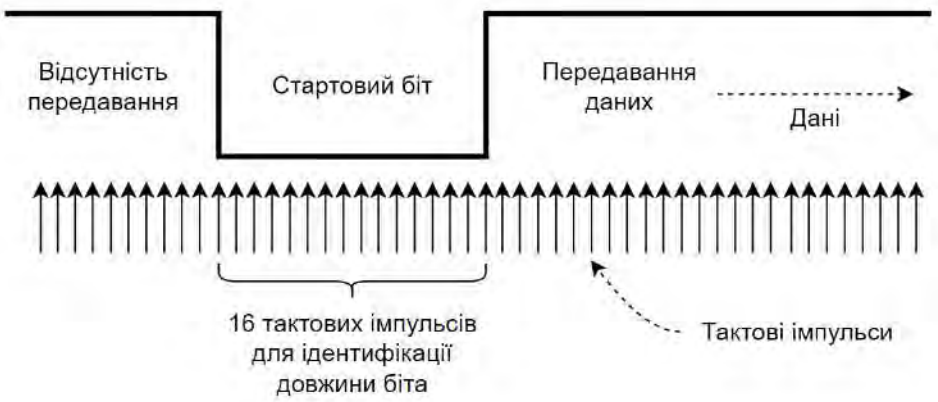


Рисунок 2.7 – Вилучення синхросигналу

2.2.4 Принцип реалізації синхронної послідовної передачі даних

У синхронних протоколах між символами, що пересилаються (байтами), немає стартових і стопових сигналів, тому окремі символи в цих протоколах пересилати не можна. Усі обміни даними здійснюються кадрами, які мають у загальному випадку заголовок, поле даних та ознаку кінця (рис. 2.8). Усі біти кадру передаються безперервним синхронним потоком, що прискорює передачу даних.

Рисунок 2.8 – Кадри синхронних протоколів

Так як байти в цих протоколах не відокремлюються один від одного службовими сигналами, то одним із перших завдань приймача є розпізнавання границі байту. Потім приймач повинен знайти початок і кінець кадру, а також визначити межі кожного поля кадру – адреси призначення, адреси джерела, інших службових полів заголовка, поля даних та контрольної суми, якщо вона є. Більшість протоколів допускає використання кадру поля даних змінної довжини.

Зазвичай, протоколи визначають максимальне значення, яке може мати довжина поля даних. Ця величина називається максимальною одиницею передачі даних (Maximum Transfer Unit, MTU).

Синхронні протоколи канального рівня бувають двох типів:

– символьно-орієнтовані (байт-орієнтовані);

– біт-орієнтовані.

Для обох типів характерні одні й ті ж самі методи синхронізації бітів. Головна різниця між ними полягає у методі синхронізації символів та кадрів.

Символьно-орієнтовані протоколи використовуються в основному для передачі блоків символів, що відображаються, наприклад текстових файлів. Оскільки в синхронній передачі немає стопових і стартових бітів, для синхронізації символів необхідний інший метод. Синхронізація досягається за

рахунок того, що передавач додає два або більше керуючих символів, які називаються символами SYN, перед кожним блоком символів.

Приклад структури такого повідомлення подано на рису. 2.9.

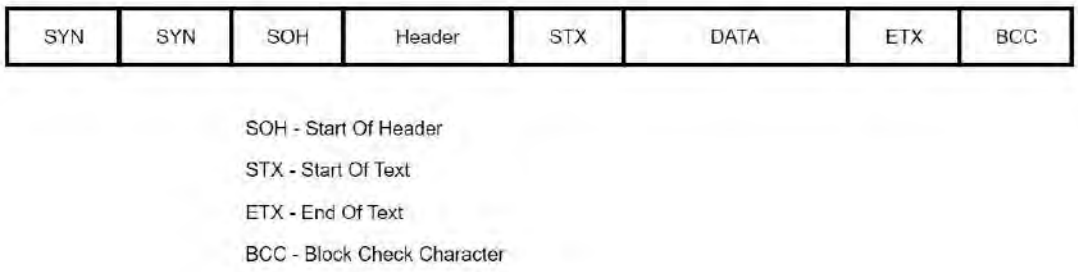


Рисунок 2.9 – Структура кадру передачі інформації за протоколом BSC

Символи SYN виконують дві функції: по-перше, вони забезпечують приймачеві бітову синхронізацію, по-друге, як тільки бітова синхронізація досягається, вони дозволяють приймачеві розпочати розпізнавання меж символів SYN.

Після того як приймач почав відокремлювати один символ від іншого, можна встановити межі початку кадру за допомогою іншого спеціального символу. Зазвичай у символьних протоколах для цих цілей використовується символ STX . Інший символ відзначає закінчення кадру – ETX.

Найбільш популярним протоколом такого типу був протокол BSC компанії IBM (рис. 2.9). Він працював у двох режимах – непрозорому, у якому деякі спеціальні символи всередині кадру заборонялися, та прозорому, в якому дозволялася передачі всередині кадру будь-яких символів, у тому числі ETX.

Потреба в парі символів на початку і наприкінці кожного кадру разом із додатковими символами DLE означає, що символьно-орієнтована передача не ефективна для передачі двійкових даних, оскільки доводиться у полі даних кадру додавати чимало надлишкових даних.

Щоб подолати ці проблеми, сьогодні майже завжди використовується більш універсальний метод, який називається *біт-орієнтованою передачею*. Цей метод зараз застосовується під час передачі як двійкових, так і символьних даних. На рис. 2.10 подано три різні схеми біт-орієнтованої передачі. Вони відрізняються способом позначення початку та кінця кожного кадру.

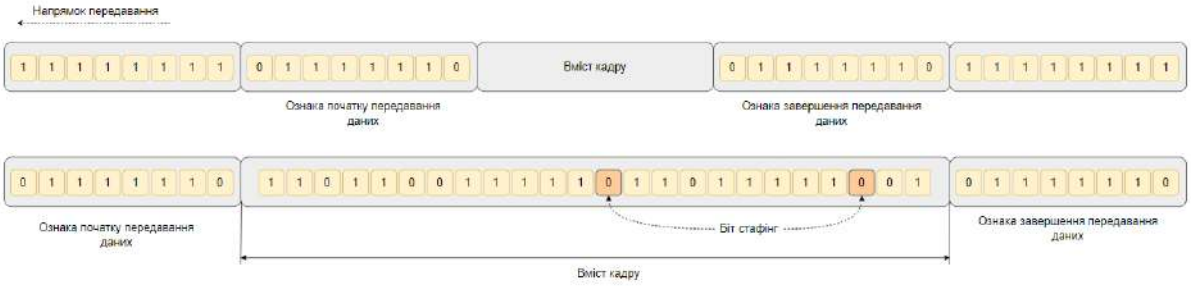


Рисунок 2.10 – Способи виділення початку та кінця кадру в синхронній передачі

При передачі кадрів даних на канальному рівні використовуються як дейтаграмні процедури, що працюють без встановлення з'єднання (connectionless), так і процедури з попереднім встановленням логічного з'єднання (connection-oriented).

В дейтаграмній передачі кадр надсилається до мережі «без попередження», і жодної відповідальності за його втрату протокол не несе (рис. 2.10). Передбачається, що мережа завжди готова прийняти кадр від кінцевого вузла. Дейтаграмний метод працює швидко, оскільки жодних попередніх дій перед надсиланням даних не потрібно. Однак за такого методу важко організувати у межах протоколу відстеження факту доставки кадру вузла призначення. Цей метод не гарантує доставки пакета.

Схема подана на рис. 2.10, схожа на схему з символами STX і ETX в байт-орієнтованих протоколах. Початок і кінець кожного кадру відзначені однією й тією ж 8-бітовою послідовністю – 01111110, яка називається прапором. Термін «біторієнтований» використовується тому, що потік бітів, що приймається, сканується приймачем на побітовій основі для виявлення стартового прапора, а потім під час прийому для виявлення стопового прапора. Тому довжина кадру в такому разі не обов'язково має бути кратна байту.

Щоб забезпечити синхронізацію приймача, передавач посилає послідовність байтів простою (11111111), що передує стартовому прапору. Для досягнення прозорості даних у цій схемі необхідно, щоб прапор не зустрічався у полі даних кадру. Це досягається за допомогою прийому, відомого як вставка 0-го біта – бітстафінгу.

Схема вставки біта працює лише під час передачі поля даних кадру. Якщо ця схема виявляє, що підряд передано п'ять одиниць, вона у будь-якому разі

автоматично вставляє додатковий нуль. Тому послідовність 01111110 ніколи не з'явиться у полі даних кадру. Аналогічна схема працює у приймачі і виконує зворотну функцію. Коли після п'яти одиниць виявляється нуль, він автоматично видаляється із поля даних кадру. Бітстафінг економніше байтстафінгу, оскільки замість зайвого байту вставляється один біт, отже, швидкість передачі даних знижується повільніше.

На рис. 2.11 подано приклад реалізації методу передачі зі встановленням з’єднання.

Рисунок 2.11 – Приклад реалізації методу передачі зі встановленням з’єднання

Передача із встановленням з'єднання більш надійна, але потребує більше часу передачі даних і обчислювальних витрат від кінцевих вузлів. У цьому випадку вузлу-одержувачу надсилається службовий кадр спеціального формату з пропозицією встановити з'єднання (рис. 2.11). В даній схемі для позначення початку кадру передбачений тільки стартовий прапор, а для визначення кінця кадру використовується поле довжини кадру, яке при фіксованих розмірах заголовка і кінцевого поля найчастіше має сенс довжини поля даних кадру.

Ця схема найбільш застосовна у локальних мережах, у яких позначення факту незайнятості середовища – це взагалі відсутність передачі жодних символів. Щоб усі інші станції увійшли в бітову синхронізацію, станція, що посилає, передує вміст кадру послідовністю біт, відомою як преамбула, яка складається з чергування одиниць і нулів 101010...

Увійшовши в бітову синхронізацію, приймач досліджує вхідний потік на побітовій основі, поки не виявить байт початку кадру 10101011, що виконує роль символу STX. За цим байтом слідує заголовок кадру, в якому в певному місці знаходиться поле довжини поля даних. Таким чином, у цій схемі приймач просто відраховує задану кількість байт, щоб визначити закінчення кадру.

Схема, подана на рис. 2.12, використовує для позначення початку і кінця кадру прапори, які включають заборонені для цього коду сигнали (Code Violations, V).

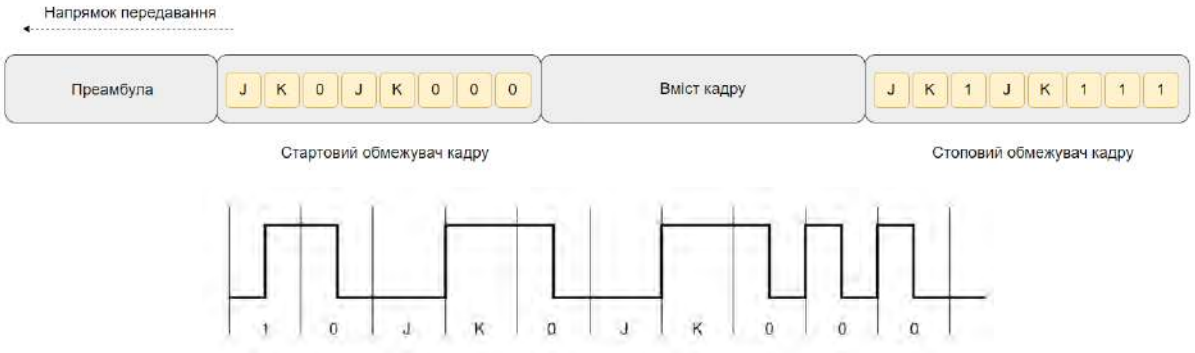


Рисунок 2.12 – Використання спеціальних символів для позначення початку і кінця кадру

Наприклад, при манчестерському кодуванні замість обов'язкової зміни полярності сигналу в середині тактового інтервалу рівень сигналу залишається незмінним і низьким (заборонений сигнал J) або незмінним і високим (заборонений сигнал K).

Початок кадру відзначається послідовністю JK0JK000, а кінець – послідовністю JK1JK111. Цей спосіб дуже економічний, тому що не вимагає ні бітстафінгу, ні поля довжини. Недоліком цього є те, що він залежить від прийнятого методу фізичного кодування.

Якщо використовуються надлишкові коди, роль сигналів J і K грають заборонені символи, наприклад, цими символами можуть бути коди 11000 і 10001.

Кожна з трьох схем має свої переваги і недоліки. Прапори дозволяють відмовитися від спеціального додаткового поля, але вимагають спеціальних заходів: або з дозволу розміщення прапора в полі даних за рахунок бітстафінгу, або використання в якості прапора заборонених сигналів, що робить цю схему залежною від способу кодування.

2.3 Загальна характеристика основних промислових протоколів

2.3.1 Мережна архітектура багаторівневої системи управління

До параметрів контролерів, що характеризує їх здатність взаємодіяти з іншими пристроями системи управління, відносяться [12]:

– кількість та різноманітність портів у процесорних модулях;

– широта набору інтерфейсних модулів та інтерфейсних процесорів;

– протоколи, що підтримуються;

– швидкість обміну даними та протяжність каналів зв'язку.

На рис. 2.13 подана мережна архітектура багаторівневої системи управління [12].

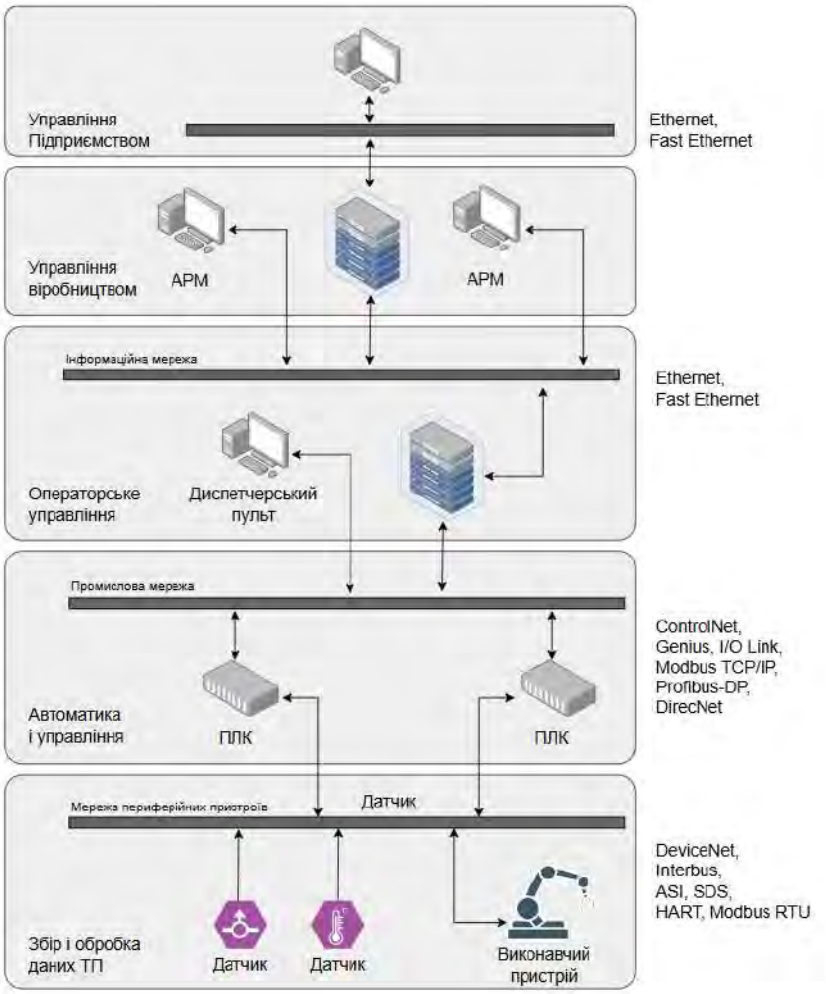


Рисунок 2.13 – Мережна архітектура багаторівневої системи управління

Пристрої верхнього рівня (комп'ютери, концентратори) на своєму рівні обмінюються великими обсягами інформації. Ця інформація захищена механізмами підтверджень та повторів на рівні протоколів взаємодії. Масив даних, що пересилається, може бути доступний не тільки центральному пристрою, але і іншим вузлам мережі цього рівня. Це означає, що мережа є рівноправною (одноранговою), тобто визначається моделлю взаємодії

peer-to-peer (рівний з рівним). Час доставки інформації не є домінуючою вимогою до цієї мережі (йдеться про жорсткий реальний час).

Мережі, які забезпечують інформаційний обмін на цьому рівні, називають інформаційними мережами. Найбільш яскравим представником мереж цього рівня є Ethernet із протоколом TCP/IP.

Мережі, які забезпечують інформаційний обмін між контролерами, датчиками і виконавчими пристроями, часто об'єднуються під загальною назвою – промислові мережі.

Промислові мережі можна поділити на два рівні:

– керуючі промислові мережі, які мають вирішальні завдання збору та обробки даних на рівні промислових контролерів, управління технологічним процесом;

– польові мережі або шини, завдання яких зводяться до опитування датчиків та управління роботою різноманітних виконавчих пристроїв.

Для забезпечення безпомилковості та максимальної зручності передачі інформації, мережні операції регулюються набором правил та угод, які називають мережним протоколом. Мережний протокол визначає типи роз'ємів, кабелів, сигнали, формати даних та способи перевірки помилок, а також алгоритми для мережних інтерфейсів та вузлів, припускаючи стандартними в межах мережі принципи підготовки повідомлень та їх передачі.

На сьогоднішній день спектр протоколів для обох цих класів промислових мереж (керівні та польові) досить широкий.

CAN, FIP, Profibus, ControlNet, DH+, Modbus, Modbus plus, Genius, DirectNet, DeviceNet, Interbus, SDS, ASI, HART, FF та ще кілька десятків протоколів присутні сьогодні на ринку промислових мереж. Кожна з мереж має свої особливості та сфери застосування.

2.3.2 Протокол Modbus

Для організації взаємодії між елементами автоматизації у промислових мережах передачі широко застосовується комунікаційний протокол Modbus. Протокол Modbus можна назвати найпоширенішим у світі (рис. 2.14).

Існують три основні реалізації протоколу Modbus, дві для передачі даних послідовними лініями зв'язку, як мідним EIA/TIA-232-E (RS-232), EIA-422, EIA/TIA-485-A (RS-485), так і оптичним, і радіо: Modbus RTU і Modbus ASCII, передача даних по мережах Ethernet поверх TCP/IP: Modbus TCP.

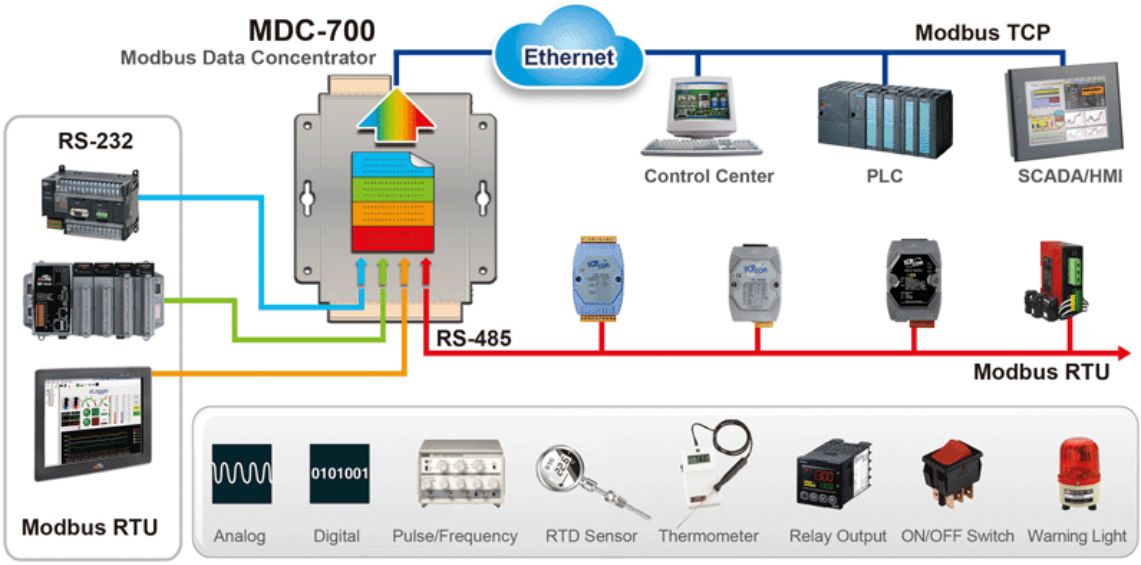


Рисунок 2.14 – Сумісність протоколів сімейства Modbus

Відмінність між протоколами Modbus ASCII та Modbus RTU полягає у способі кодування символів. У режимі ASCII дані кодуються за допомогою таблиці ASCII, де кожному символу відповідає два байти даних. У режимі RTU дані передаються у вигляді 8-розрядних двійкових символів, що забезпечує більш високу швидкість передачі даних. ASCII допускає затримку до 1 секунди на відміну від RTU, де повідомлення мають бути безперервними. Також режим ASCII має спрощену систему декодування та управління даними.

Протоколи сімейства Modbus (Modbus ASCII, Modbus RTU та Modbus TCP/IP) використовують один прикладний протокол, що дозволяє забезпечити їхню сумісність. Максимальна кількість мережних вузлів у мережі Modbus – 31. Протяжність ліній зв'язку та швидкість передачі залежить від фізичної реалізації інтерфейсу. Елементи мережі Modbus взаємодіють, використовуючи клієнт-серверну модель, засновану на транзакціях, що складаються із запиту та відповіді.

Зазвичай у мережі є лише один клієнт, так званий, «головний» (англ. master) пристрій, і кілька серверів – «підлеглих» (slaves) пристроїв. Головний пристрій ініціює транзакції (передає запити). Підлеглі пристрої передають дані, що запитуються головним пристроєм, або роблять запитані дії. Головний може адресуватися індивідуально до підлеглого або ініціювати передачу широкомовного повідомлення всім підлеглим пристроям. Підлеглий пристрій формує повідомлення та повертає його у відповідь на запит, адресований саме йому.

Області промислового застосування: організація зв'язку датчиків та виконавчих механізмів з контролером, зв'язок контролерів та керуючих комп'ютерів, зв'язок з датчиками, контролерами та корпоративними мережами, у SCADA системах.

Простота застосування протоколів сімейства Modbus у промисловості зумовило його широке поширення. На сьогоднішній день обладнання практично всіх виробників підтримує протоколи Modbus.

Традиційно протоколи сімейства Modbus підтримуються OPC серверами SCADA систем (Clear SCADA, Control Microsystems, InTouch Wonderware, TRACE MODE) для зв'язку з елементами управління (контролерами, ЧРП, регуляторами тощо).

Усі функції, які підтримує протокол Modbus, розпізнаються за ідентифікаційним номером. Функції використовуються як керуючі команди для польових вимірювальних приладів і виконавчих механізмів. До цих функцій належать:

– команди управління котушками для читання стану та встановлення однієї або групи котушок;

– команди керування входами для читання стану одного чи групи входів;

– команди управління регістрами для читання та встановлення одного або кількох регістрів тимчасового зберігання інформації;

– функції діагностики та звітів;

– функції керування опитуванням;

– функція скидання.

2.3.3 CANbus

CANbus (Control Area Network) – це послідовна шина з децентралізованим доступом.

Протокол CANbus закриває 1-й та 2-й рівні моделі OSI [13]. За своїми характеристиками він задовольняє як вимогам завдань реального часу, так й реалізує високий рівень виявлення та виправлення помилок. У кожному повідомленні може бути надіслано до 8 біт даних. Великі блоки можна передавати за допомогою використання принципу сегментації.

Можливі колізії, пов'язані з одночасним запитом шини, дозволяються на основі пріоритетності повідомлень, що передаються. У CANbus кожен блок даних містить додатковий 11-бітовий ідентифікатор, який є пріоритетом даного

повідомлення. Право працювати з шиною отримає той вузол, який передає повідомлення з найвищим пріоритетом.

Рисунок 2.15 – Використання протоколу Modbus

CAN характеризується такими основними властивостями:

– кожному повідомленню (а не пристрою) встановлюється свій пріоритет;

– гарантована величина паузи між двома актами обміну;

– гнучкість конфігурування та можливість модернізації системи;

– широкомовний прийом повідомлень із синхронізацією часу;

– несуперечність даних на рівні всієї системи;

– допустимість кількох провідних пристроїв у мережі («багатомайстерна мережа»);

– здатність до виявлення помилок та сигналізації про їх наявність;

– автоматичний повтор передачі повідомлень, доставлених з помилкою, відразу, як мережа стане вільною;

– автоматичне розрізнення збоїв і відмов з можливістю автоматичного відключення модулів, що відмовили.

До недоліків можна віднести порівняно високу вартість CAN-пристроїв, відсутність єдиного протоколу прикладного рівня, а також надмірну складність

та заплутаність протоколів канального та прикладного рівня, викладених у стандартах організації CAN in Automation (CiA).

Кожен модуль у мережі має CAN-контролер і CAN-трансивер на CAN-чіпі. Трансивер має можливість як передавати, так і отримувати дані. Контролер CAN перетворює дані (двійкові), надіслані з мікропроцесора, і надсилає їх на трансивер CAN. Трансивер CAN перетворює двійкові дані в діапазон напруги, і це напруга сигналу, що спостерігається в мережі (рис. 2.16).

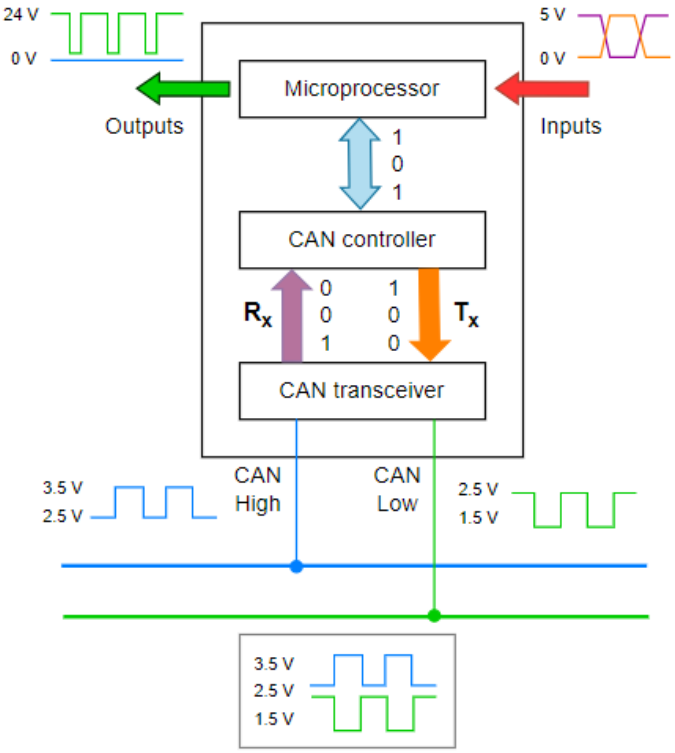


Рисунок 2.16 – Принцип підключення модулю CAN до мережі

З подачею рівня логічного нуля на вхід (вхід є інвертуючим) обидва транзистори вихідного каскаду передавача відкриваються і через навантаження (два резистори по 120 Ом) тече струм, що створює стан, відповідний логічній одиниці. При цьому потенціал виведення CAN_H завжди буде вищим, ніж виводу CAN_L (рис. 2.16). Якщо на вході передавача буде логічна одиниця, його вихід переходить у високоомний стан і диференціальна напруга на лінії дорівнюватиме нулю.

Зазначимо, що наявність термінальних резисторів в CAN необхідна не тільки для узгодження лінії, але навіть для створення шляху струму.

CAN передавач має дуже важливу властивість: якщо один із передавачів встановлює в мережі логічний нуль, а другий – логічну одиницю, то цей стан не є аварійним, оскільки наскрізного струму не виникає. У даному випадку CAN лінія залишається у стані логічної одиниці. Інакшими словами, логічна одиниця завжди домінує над логічним нулем. Тому в стандарті CAN використовується поняття «домінантний стан» стан лінії для позначення стану лінії зі струмом, та поняття «рецесивний стан», як протилежне домінантному (рис. 2.17).

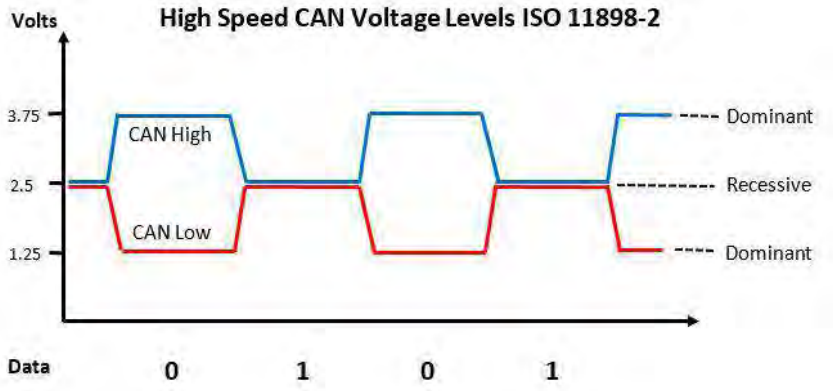


Рисунок 2.17 – Пояснення понять рецесивного та домінантного стану

Ця властивість CAN забезпечує можливість отримання доступу до лінії, порівнюючи логічні рівні, що посилаються в лінію, з тим рівнем, який фактично встановлюється в ній: якщо передавач посилає в лінію рецесивний стан, а в ній при цьому залишається домінантний, значить лінія зайнята.

Доступ отримує той вузол мережі, який може надати домінантний рівень сигналу. Вузли з рецесивним рівнем залишають лінію і чекають на наступний випадок. Цей метод доступу справедливий і для використання оптоволоконного каналу або бездротової мережі – у цих випадках наявність світла або електромагнітної хвилі завжди домінуватиме над їх відсутністю.

2.3.4 Протокол Profibus

Протокол Profibus (PROcess FIeld BUS) розроблено у Німеччині. Стандарт протоколу описує рівні 1, 2 та 7 OSI-моделі. У Profibus використовується гібридний метод доступу Master/Slave та децентралізована процедура передачі маркера. Мережа може складатися зі 122 вузлів, з яких 32 можуть бути Master-вузлами. Адреса 0 зарезервована для режиму широкого мовлення. У середовищі Master-вузлів за зростаючими номерами вузлів передається маркер, який надає

право ведення циклів читання/запису на шині. Усі цикли суворо регламентовані за часом, організовано продуману систему тайм-аутів. Протокол добре дозволяє різноманітні колізії на шині. Налаштування всіх основних часових параметрів відбувається за сценарієм користувача. Робоча швидкість передачі може бути обрана в діапазоні 9,6-12 000 кбіт/с.

В процесі побудови багаторівневих систем автоматизації часто виникають завдання організації інформаційного обміну рівнями. В одному випадку потрібний обмін комплексними повідомленнями на середніх швидкостях. В іншому – швидкий обмін короткими повідомленнями з використанням спрощеного протоколу обміну (рівень датчиків). У третьому потрібна робота у небезпечних ділянках виробництва (нафтогазові технології, хімічне виробництво). Для всіх цих випадків Profibus має рішення. Під загальною назвою розуміється сукупність трьох окремих протоколів: Profibus-FMS, Profibus-DP та Profibus-PA.

Протокол Profibus-FMS з'явився першим і призначений для роботи на так званому цеховому рівні. Тут потрібен високий ступінь функціональності, і цей критерій важливіший за критерій швидкості. Основне призначення – передача великих обсягів даних.

У завданнях управління, потребують реального часу, перше місце висувається такий параметр, як тривалість циклу шини. Реалізація протоколу Profibus-DP дає збільшення продуктивності шини (наприклад, для передачі 512 біт даних, розподілених по 32 станціям, потрібно всього 6 мс).

Протокол Profibus-PA – це розширення DP-протоколу в частині технології передачі, що засновано не на RS-485, а на реалізації стандарту IEC1158-2 для організації передачі у вибухонебезпечних середовищах. Він може використовуватися як заміна старої аналогової технології 4-20 мА. Для комутації пристроїв потрібна лише одна кручена пара, яка може одночасно використовуватися і для інформаційного обміну, і для підведення живлення до пристроїв польового рівня.

Протокол Profibus-DP підтримується пристроями різних виробників. Для контролерів Siemens цей протокол є основним (рис. 2.18). Деякі контролери сімейств S7-300 та S7-400 мають вбудований порт Profibus-DP, інші взаємодіють із мережею за допомогою комунікаційних процесорів.

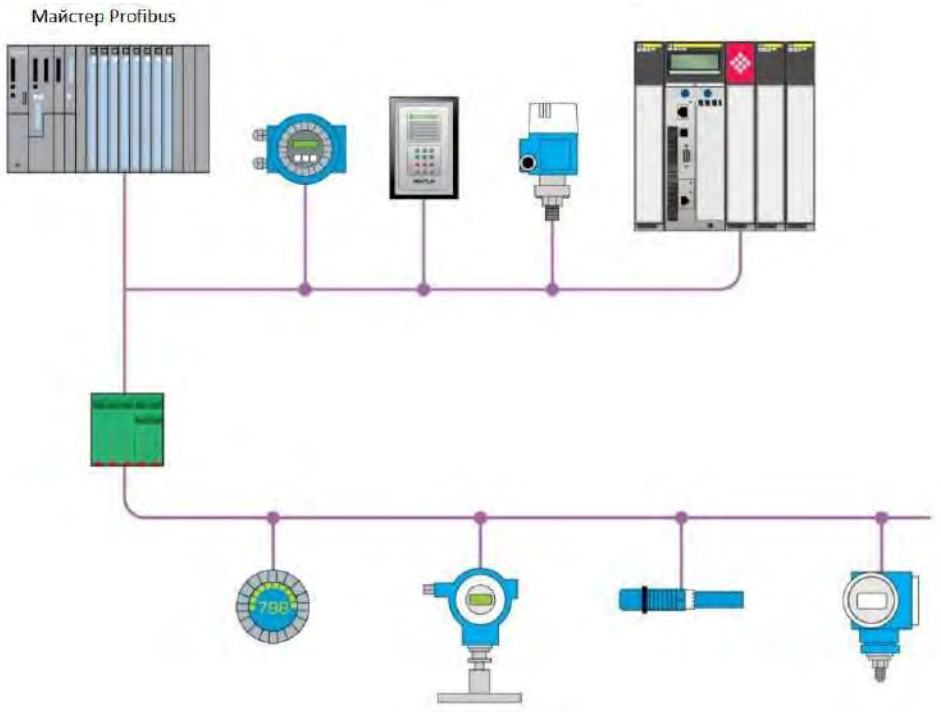


Рисунок 2.18 – Мережа Profibus

2.3.5 Взаємодія з пристроями польового рівня

В останні роки з’явилася тенденція застосування в системах управління технологій наскрізного мережного доступу: від потужних супервізорних комп'ютерів та багатофункціональних контролерів до інтелектуальних польових пристроїв (датчики, виконавчі пристрої тощо). При цьому, такий зв'язок має задовольняти всім сучасним вимогам щодо функціональності, надійності та відкритості. Розглянуті вище мережі та протоколи не призначені для безпосередньої взаємодії з пристроями польового рівня.

Польові шини (шини рівня датчиків та виконавчих пристроїв) повинні відповідати двом вимогам. По-перше, необхідно передавати дані відповідно до жорсткого тимчасового регламенту. По-друге, обсяг даних має бути мінімальним, щоб забезпечити працездатність мережі в критичні за навантаженнями моменти. Мережа рівня датчиків забезпечує безпосередній інтерфейс між реальним технологічним процесом та промисловими контролерами.

Інформацію, що передається такою мережею, можна розділити на два основні типи: дані про процес і параметричні дані. Обидва типи даних принципово різні і пред'являють до комунікаційної системи різні вимоги.

Дані про процес (зміна стану кранів, перемикачів, сигналів, що управляють тощо) не є складними і, як правило, визначаються кількома інформаційними бітами. Обсяг такої інформації має чітку тенденцію до скорочення. Нещодавно ці дані для одного простого пристрою займали 8-16 біт. Але вже зараз розвиток технології призвело до того, що з найпростіших датчиків (дискретного типу) надходить лише 1-2 біти інформації.

Дані про процес мають явно виражений циклічний характер. Більше того, для реалізації завдань автоматичного керування необхідно, щоб опитування каналів та видача команд на керування проводились через регламентовані інтервали часу. Це так звана вимога детермінованості комунікаційної системи. Завдяки невеликому об'єму даних системи промислового зв'язку, що передаються, здатні дійсно задовольняти тимчасовим вимогам з боку реальних процесів.

Параметричні дані необхідні як відображення поточного стану мережних пристроїв (інтелектуальних) і їх перепрограмування. На противагу даним про процес, параметрична інформація не має циклічного характеру. Доступ до неї реалізується на запит, в ациклічному режимі. Передача параметричних даних вимагає та реалізує методи спеціального захисту, а також механізмів підтверджень. Комплексний параметричний блок для інтелектуальних пристроїв займає від кількох десятків байт до кількох сотень кілобайт. У порівнянні з даними, що швидко змінюються, тимчасові вимоги до передачі параметрів можна вважати некритичними. Залежно від типу пристроїв та протяжності мережі, вимоги по часу простягаються від кількох сотень мілісекунд до кількох хвилин. Розглянемо кілька промислових шин рівня датчиків та виконавчих пристроїв (польових шин), які успішно застосовуються в автоматизації технологічних процесів.

2.3.6 Actuator Sensor Interface

Технологія AS-Interface (англ. Actuator Sensor Interface) підтримується низкою відомих фірм: Allen-Bradley, Siemens, Schneider Electric та іншими (рис. 2.19).

Основне завдання цієї мережі – зв'язати в єдину інформаційну структуру пристрою нижнього рівня автоматизованого процесу (фотоелектричні датчики, виконавчі пристрої, реле, контактори, ємнісні перемикачі, приводи тощо)

із системою контролерів. Це підтверджується і назвою мережі Actuator Sensor Interface.

AS-Interface дозволяє через свої комунікаційні лінії не тільки передавати дані, а й підводити живлення (24 В DC) до датчиків та виконавчих пристроїв. Тут використовується принцип послідовної передачі на базовій частоті. Інформаційний сигнал модулюється на частоту живлення.

Рисунок 2.19 – Модулі A&D SIEMENS провідних пристроїв AS-Interface

Топологією ASI-мережі може бути шина, зірка, кільце або дерево. До одного контролера можна підключити до 31 пристроїв. Протяжність сегмента ASI-шини може досягати 100 м. За рахунок повторювачів довжину мережі та кількість вузлів можна збільшувати. Цикл опитування 31 вузла – до 5 мс. Максимальний обсяг даних з одного ASI-вузла – 4 біта.

Тенденція у побудові розподілених систем автоматизації має явне прагнення використовувати технології наскрізного доступу до мережі. За допомогою мережі AS-Interface можна будувати системи, в яких датчики та контролери пов'язані однією мережею. AS-Interface має шлюзи в інші промислові мережі: Profibus, Interbus-S тощо.

Кожен вузол ASI-мережі повинен мати спеціальний інтерфейсний контролер із підтримкою ASI-протоколу. AS-Interface дозволяє передавати як дані, так і живильне навантаження до вузлів мережі, оскільки існує велика кількість фотоелектричних та індуктивних датчиків.

Логічним центром будь-якої топології є MASTER-вузол, який контролює всю роботу мережі, організовує обмін даними з ПЛК. ASI-MASTER може бути організований в широкому спектрі контролерів, якими організуються шлюзи

у промислові мережі вищого рівня. Часто ASI-MASTER оформлюється як окрема плата контролера чи комп'ютера. Максимальна кількість вузлів до одного MASTER-вузла – 31.

Як середовище передачі використовується пара звичайних провідників. Швидкість передачі обмежена до 167 кбод. Сьогодні з'явився спеціальний ASI-кабель, в якому обидва провідники упаковані в спеціальну м'яку гумову оболонку, яка робить цей кабель гнучким та стійким до багаторазових вигинів.

Для кодування даних використовується відомий Манчестерський код, в якому «0» і «1» кодуються по висхідному та низхідному фронтові сигналу. Такий тип кодування знижує вплив на ASI-кабель зовнішніх збурень.

Приклад структури мережі пристроїв, що поєднані AS-Interface подано на рис. 2.20.

Компоненти системи для мережі AS-Interface:

– головний пристрій AS-Interface;

– підлеглі пристрої AS-Interface, які, залежно від конструкції, поділяються на модулі AS-Interface, датчики / виконавчі механізми із вбудованим AS-Interface;

– кабель AS-Interface;

– блок живлення AS-Interface;

– прилад для завдання адрес;

– програма SCOPE для AS-Interface.

Відмінними рисами AS-Interface є такі основні характеристики:

– AS-Interface оптимальний для підключення бінарних датчиків та виконавчих механізмів. Кабель AS-Interface використовується як для обміну даними між датчиками / виконавчими механізмами (підлеглими пристроями AS-Interface) та головним пристроєм AS-Interface, так і для подавання напруги живлення на датчики / виконавчі механізми;

– простіший та економічніший монтаж з'єднань. Завдяки використанню методу проколювання ізоляції спрощується монтаж кабелю та досягається висока гнучкість, необхідна для побудови деревоподібної топології;

– малий час реакції: головному пристрою AS-Interface потрібно не більше 5 мс для циклічного обміну даними з 31 вузлом мережі;

– як вузли (AS-Interface підлеглих) кабелю AS-Interface можуть виступати або датчики / виконавчі механізми з вбудованим AS-Interface, або модулі

AS-Interface, до яких можна підключити до 4 звичайних бінарних датчиків / виконавчих механізмів.

– під час використання стандартних AS-Interface модулів на кабелі AS-Interface може бути до 124 виконавчих механізмів / датчиків;

– якщо використовуються AS-Interface модулі з розширеним режимом адресації, один головний пристрій з розширеним режимом адресації може працювати до 186 виконавчих механізмів і 248 датчиків;

– розширені головні пристрої AS-Interface сімейства SIMATIC NET забезпечують надзвичайно простий доступ до аналогових датчиків / виконавчих механізмів або модулів, функціонування яких відповідає профілю підлеглих пристроїв AS-Interface.

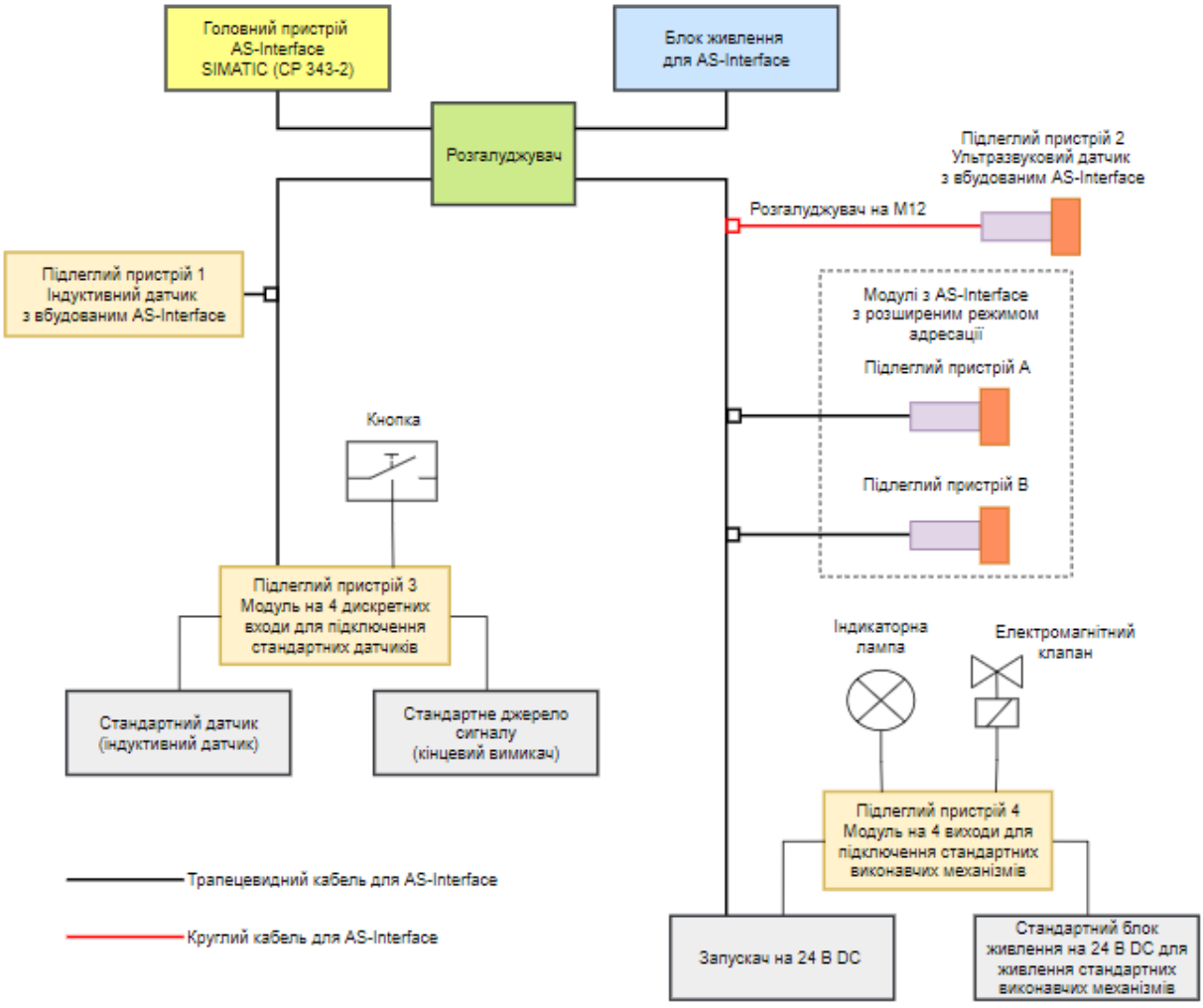


Рисунок 2.20 – Приклад структури мережі пристроїв, що поєднані AS-Interface

2.3.7 HART протокол та інтелектуальні вимірювальні прилади

Протоколи інтелектуальних вимірювальних приладів розроблені для застосування, в яких реальні дані збираються вимірювальними приладами, датчиками та виконавчими механізмами з використанням цифрового каналу зв'язку. Всі ці пристрої безпосередньо підключаються до програмованих логічних контролерів (ПЛК) та комп'ютерів.

Протокол HART (дистанційний магістральний перетворювач, що адресується) є типовою реалізацією шини інтелектуальних вимірювальних приладів Fieldbus, яка може працювати в гібридному цифровому режимі зі струмовою петлею 4-20 мА (рис. 2.21).

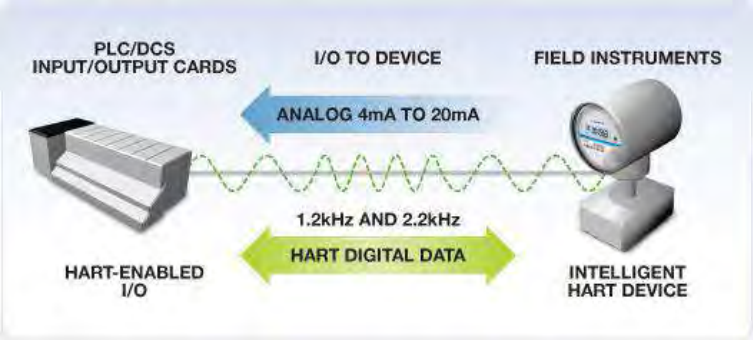


Рисунок 2.21 – Поєднання пристроїв за допомогою HART протоколу

Протокол HART у жодному разі не є єдиним протоколом у цій сфері. Існують сотні інтелектуальних протоколів різних виробників – наприклад, Honeywell, який сумісний із HART.

На базовому рівні більшість інтелектуальних приладів забезпечує такі функції:

– контроль за вибором діапазону, налаштування нуля та інтервалу вимірювань;

– діагностика функціонування;

– збереження конфігурації та інформації про стан приладу (наприклад, кодові мітки тощо).

Доступ до цих функцій забезпечує значне збільшення швидкості та ефективності процесу налаштування та експлуатації системи. Наприклад, на діагностику повільної струмової петлі 4-20 мА може йти кілька хвилин, протягом цього часу пристрій готується до вимірювань шляхом налаштування нульових значень, а також інших параметрів типу коефіцієнтів демпфування.

Спочатку HART протокол був розроблений компанією Rosemaunt, а тепер є відкритим стандартом, доступним всім виробникам. Його головною перевагою є те, що він дозволяє зберегти наявні кабельні з'єднання вимірювальних приладів, необхідні для струмової петлі 4-20 мА, і використовувати їх для передачі цифрової інформації, накладеної на аналоговий сигнал. Це дозволяє більшості компаній зберегти їх вкладені кошти у існуюче кабельне господарство та пов'язані з ним системи і забезпечити сумісність із протоколом HART без великих грошових витрат.

На противагу більшості систем Fieldbus, які є суто цифровими, HART є гібридним (аналоговим та цифровим) протоколом.

Протокол HART використовує частотну модуляцію (FSK), засновану на комунікаційних стандартах BELL202. Дві окремі частоти 1200 і 2200 Гц представляють цифрові 1 і 0 відповідно. Усереднене значення синусоїдальної хвилі (на частотах 1200 і 220 Гц), що накладається на струмовий сигнал 4-20 мА, дорівнює нулю, тому на аналогову інформацію не впливає ніякого впливу.

Приклад застосування частотної модуляції для передачі інформації подано на рис. 2.22.

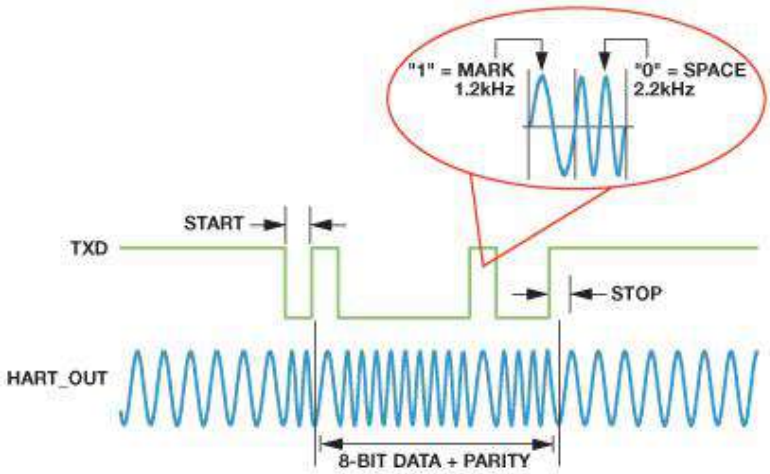


Рисунок 2.22 – Приклад застосування частотної модуляції для передачі інформації

Протокол HART можна використовувати такими способами:

– у поєднанні зі струмовим сигналом 4-20 мА у двоточковому режимі;

– у поєднанні з іншими польовими приладами у багатоточковому режимі;

– у двоточковому режимі, коли один польовий прилад передає пакети.

Звичайні двоточкові петлі, як адресу опитування інтелектуального вимірювального приладу, використовують нуль. Установка для інтелектуального пристрою адреси опитування більше за нуль створює багатоточкову петлю. Тоді, інтелектуальний пристрій встановлює на виході постійний струм 4 мА і виробляються комунікації тільки в цифровому вигляді.

Для цифрової передачі даних HART використовує два режими:

– режим опитування / відповіді;

– пакетний режим (передача інформації одразу кільком пристроям).

На рис. 2.23 подано приклад передачі інформації одразу кільком пристроям з використанням протоколу HART.

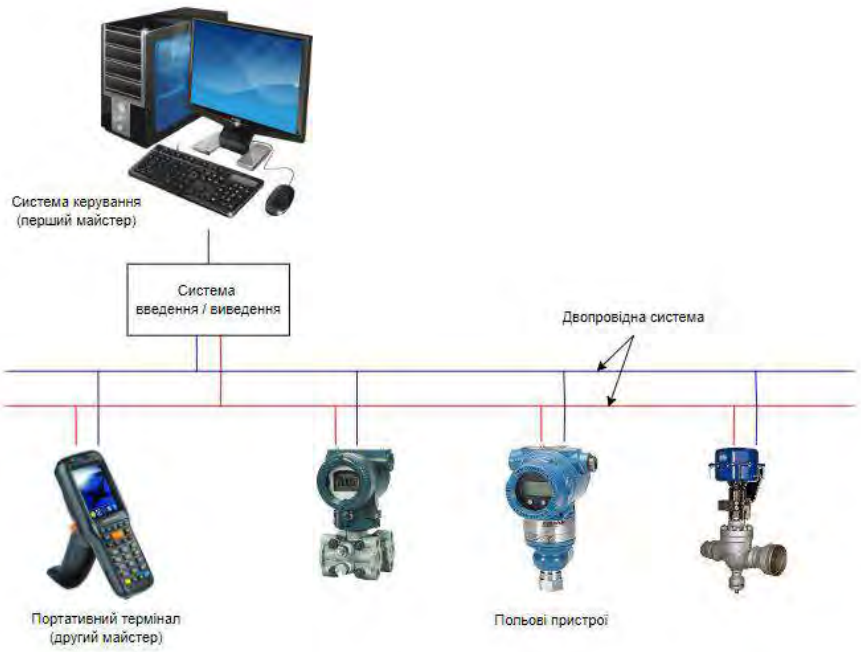


Рисунок 2.23 – Приклад передачі інформації одразу кільком пристроям з використанням протоколу HART

У режимі опитування / відповіді головний пристрій опитує кожен автономний пристрій, підключений до магістралі, і потребує передачі необхідної інформації. У пакетному режимі польовий пристрій безперервно передає дані, що стосуються процесу, при цьому головному пристрою не потрібно надсилати запити. Хоча цей режим є досить швидким (до 3,7 разів на секунду), у багатоточкових мережах його використовувати не можна.

Протокол реалізується з використанням рівнів 1, 2 та 7 моделі OSI.

HART-пристрої завжди містять мікроконтролер (рис. 2.24) з UART і ППЗП (перепрограмований постійний запам'ятовуючий пристрій). Цифровий сигнал, сформований мікроконтролером, перетворюється на UART у безперервну послідовність біт, що складається з двійкових слів завдовжки 11 біт кожне (рис. 2.25, а).

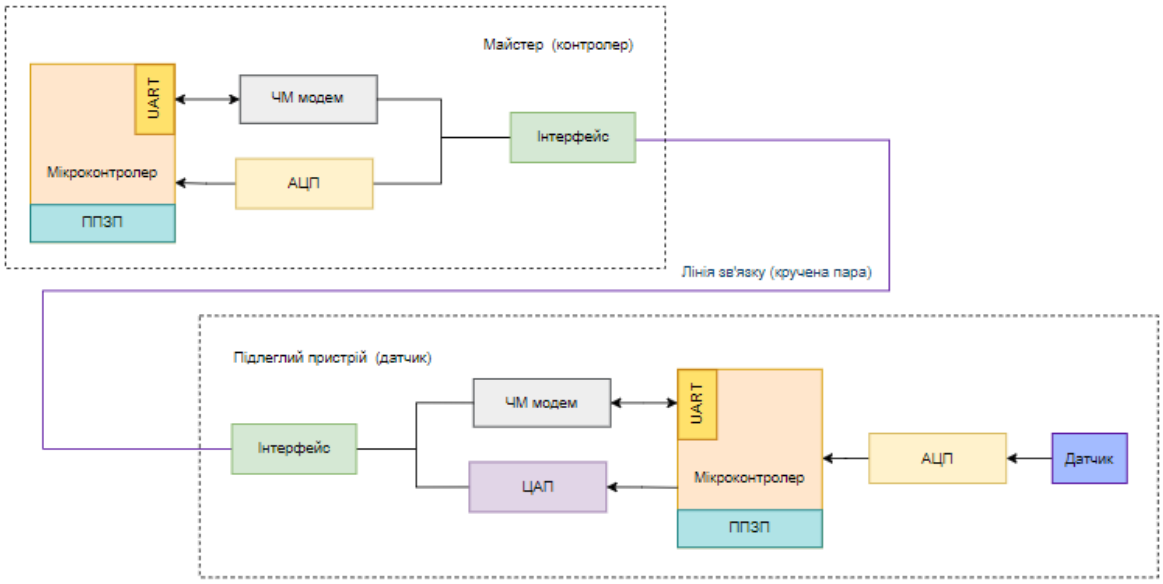


Рисунок 2.24 – Проходження аналогових та цифрових сигналів через пристрої з HART-протоколом

Кожне слово починається зі стартового біта (логічний нуль), за яким слідує байт даних, що передається, потім біт паритету і стоповий біт. Сформована в такий спосіб послідовність нулів і одиниць передається до модему, де виконується частотна маніпуляція (ЧМн). Отриманий частотно-маніпульований сигнал передається в інтерфейсний блок для формування напруги, що подається в лінію зв'язку (нагадаємо, що від контролера до датчика передається сигнал у формі напруги, а назад – у формі струму).

На стороні датчика сигнал з лінії приймається інтерфейсним блоком, перетворюється ЧС модемом на послідовність бітів, з якої контролер виділяє байти даних і біти паритету. Мікроконтролер перевіряє відповідність біта паритету переданому байту кожного переданого слова, доки виявить ознаки кінця повідомлення.

Отримавши команду, контролер вдається до її виконання. Якщо прийшла команда запиту виміряних даних, контролер датчика приймає через АЦП сигнал датчика, перетворює його в аналогову форму за допомогою ЦАП, підсумовує

службову інформацію на виході ЧС модему і передає в лінію зв'язку у формі струму 4...20 мА (рис. 2.22).

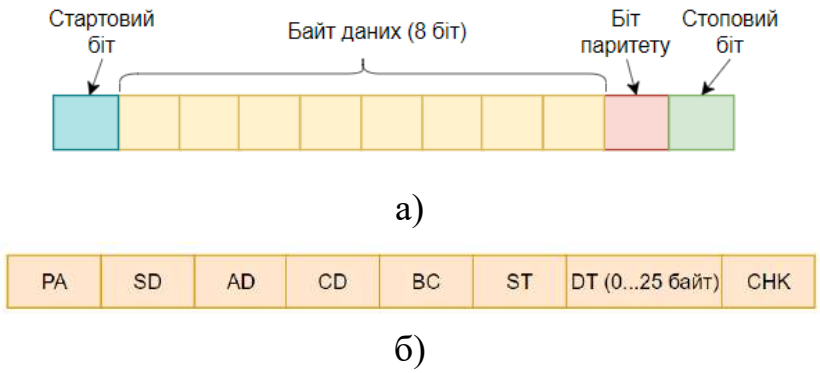


Рисунок 2.25 – Структура слова (а) та повідомлення (б) у HART-протоколі

Описаний вище обмін інформацією між двома пристроями (типу «точка – точка») є найбільш типовим застосуванням HART-протоколу. Однак HART-пристрої можуть бути об'єднані у мережу. Для цього використовують лише цифрову частину HART-протоколу, без аналогової, а інформація передається у формі напруги, що дозволяє з'єднувати HART-пристрої паралельно. Максимальна кількість пристроїв у мережі може становити 15, якщо не використовувати повторювачі HART (ретранслятори, репітери). HART-мережа може мати довільну топологію, оскільки за малих швидкостях передачі (1200 біт/с), ефектів, притаманних довгим лініям, немає. Цим пояснюються вкрай низькі вимоги до смуги пропускання кабелю (2,5 кГц за рівнем 3 дБ). Такій смузі відповідає стала часу лінії передачі 65 мкс, тобто, якщо опір лінії 250 Ом її ємність може сягати 0,26 мкФ, що відповідає довжині кабелю близько 2...3 км (табл. 2.2).

Таблиця 2.2 – Залежність довжини кабелю від погонної ємності

Кількість пристроїв в мережі	Довжина кабелю за погонної ємності, м			
	65 пФ/м	95 пФ/м	160 пФ/м	225 пФ/м
1	2800	2000	1300	1000
5	2500	1800	1150	900
10	2100	1600	1000	750
15	1800	1400	900	700

У мережі можуть бути два головні пристрої, одним з яких є контролер, другим – ручний комунікатор, який використовується для зчитування показань та встановлення параметрів HART-пристроїв. Комунікатор може бути підключений у будь-якому місці мережі, але зазвичай доступними є тільки клеми датчиків або комутаційні клеми в монтажній шафі.

Мережа допускає гарячу заміну або додавання нових пристроїв (тобто без вимкнення живлення). У разі збою, наприклад, під час випадкового короткого замикання, мережа повторює невиконані операції обміну.

У HART-мережі лише один вузол може посилати сигнал, у цей час інші слухають лінію. Ініціює процедуру обміну головний пристрій (контролер або ручний комунікатор). Підлеглі отримують команду та посилають відповідь на неї. Кожен підлеглий пристрій має персональну мережну адресу, яка включається до повідомлення головного пристрою. Адреса має довжину 4 біти («коротка адреса») або 38 біт («довга адреса»). Є також інший спосіб адресації – за допомогою тегів (ідентифікаторів, що призначаються користувачем).

2.3.8 Низькорівнева приладо-орієнтована мережа DeviceNet

DeviceNet, розроблена Allen-Bradley, є низькорівневою приладо-орієнтованою мережею на основі мережі CAN. Вона розроблена для з'єднання пристроїв нижнього рівня (датчиків та виконавчих механізмів) із пристроями високого рівня (контролерами). Для цього завдання добре підходить багатобайтний формат кадру повідомлення CAN, оскільки повідомлення такого формату передають більше інформації, ніж біторієнтовані системи.

DeviceNet – це:

– доступ до інтелектуальних датчиків різних виробників;

– зв'язок «майстер-підлеглий» та рівноправний;

– конфігурування датчиків, управління та збирання даних.

Ця мережа з'єднує пристрої нижнього рівня безпосередньо з системою керування, зменшуючи кількість зв'язків вводу / виводу та проводки по відношенню до типових апаратних рішень (рис. 2.26).

Довжина мережі DeviceNet визначається швидкістю передачі: 100 м за швидкістю 500 кбод, 200 м – 250 кбод, 500 м – 125 кбод.

DeviceNet підтримує до 64 вузлів, та до 2048 пристроїв. Живлення та передачу даних забезпечує єдиний чотирижильний кабель. Для взаємного

з'єднання пристроїв вводу / виводу є різні модулі, а налаштовувати конфігурацію дозволяє мережна магістраль.

Оскільки DeviceNet розроблена для забезпечення зв'язку пристроїв нижнього рівня з контролерами верхнього рівня, було розроблено унікальний варіант базового протоколу CAN. Він аналогічний методам опитування / відповідь або головний / підлеглий пристрій, але використовує швидкісні переваги оригінальної мережі CAN.

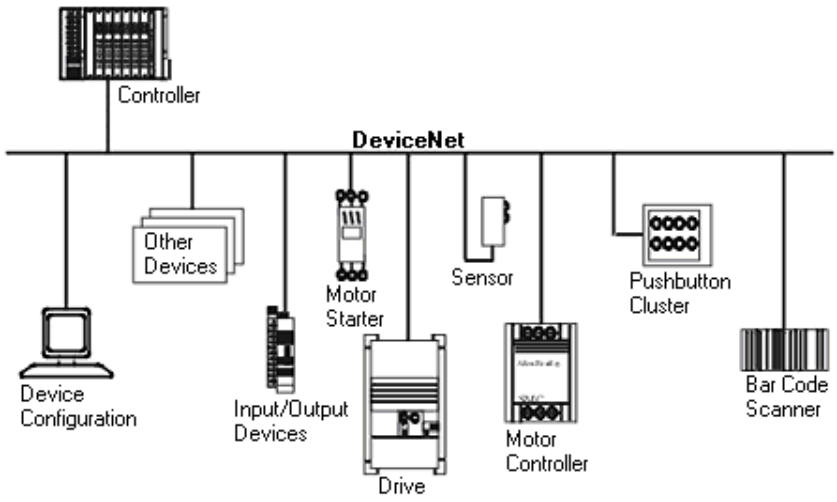


Рисунок 2.26 – Польові пристрої та модулі вводу / виводу в мережі DeviceNet

2.3.9 Industrial Ethernet та відкритий комунікаційний стандарт PROFINET

Мережа Ethernet є загальновизнаним лідером у галузі комунікаційних технологій. Вона має високу пропускну здатність, не має обмежень на кількість станцій, що підключаються, використовується в промислових і офісних умовах, забезпечує підтримку IT технологій, має безліч інших переваг. Однак у мережі Ethernet є і суттєва вада – відсутність детермінованого часу доставки повідомлень, що обмежує можливі сфери застосування цієї мережі для організації обміну даними між системами автоматизації.

Новий відкритий комунікаційний стандарт PROFINET (IEC 61158) усуває зазначені недоліки та суттєво розширює функціональні можливості обміну даними та охоплює широкий спектр вимог щодо використання Ethernet в системах автоматизації.

PROFINET орієнтовано організацію системно-широкого обміну даними між усіма ієрархічними рівнями управління підприємством. Він суттєво спрощує питання проєктування систем промислового зв'язку, поширює використання IT стандартів на польовий рівень управління, дозволяє використовувати існуючі канали зв'язку та мережні компоненти Ethernet, а також доповнювати ці мережі спеціалізованими компонентами. PROFINET забезпечує підтримку всіх існуючих стандартних механізмів обміну даними через Ethernet паралельно з обміном даними між системами автоматизації у реальному масштабі часу.

Для організації обміну даними між системами автоматизації в мережі PROFINET можуть використовуватись електричні (кручені пари), оптичні та бездротові канали зв'язку Ethernet. В залежності від виду каналів, для побудови мережі може використовуватися різний набір мережних компонентів. Забезпечується підтримка всіх топологій, притаманних мережі Industrial Ethernet: лінійних, кільцевих, деревовидних.

Для побудови мереж PROFINET, наприклад, концерн Siemens пропонує широку гаму активних та пасивних мережних компонентів, а також комунікаційного програмного забезпечення та інструментальних засобів проєктування (рис. 2.27). Більшість мережних компонентів PROFINET може використовуватись і в мережах Industrial Ethernet.

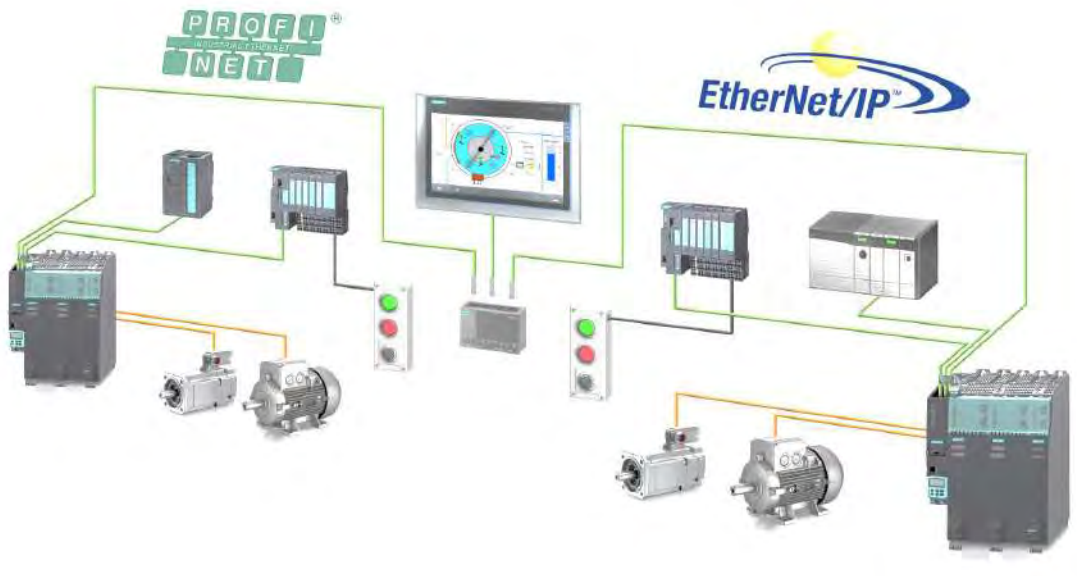


Рисунок 2.27 – Керування приладами з використанням Industrial Ethernet

Пасивні PROFINET компоненти містять в своєму складі електричні (кручені пари 2x2) та оптичні кабелі, а також з'єднувальні пристрої різного призначення. Для більшості електричних пасивних компонентів підтримується технологія FastConnect, що дозволяє виконувати швидкий та безпомилковий монтаж мережі. Усі з'єднувальні пристрої виконані з урахуванням вимог PROFINET.

Активні компоненти PROFINET представлені широкою гамою комутаторів серії SCALANCE X200/X200IRT/X300/X400. Модулі серії SCALANCE X дозволяють конфігурувати лінійні, зіркоподібні та кільцеві структури мереж Industrial Ethernet/PROFINET, використовувати для передачі даних оптичні та електричні канали зв'язку, підтримувати технологію мереж, що комутуються, дозволяють використовувати обмін даними в реальному масштабі часу, в тому числі, і з тактовою синхронізацією.

Технологічні компоненти для PROFINET представлені спеціалізованими мікросхемами ERTEC 200 та 400, а також комплектами розробки, що дозволяють спеціалістам різних фірм виконувати проєктування, макетування та налагодження інтерфейсної частини власної апаратури управління, призначеної для роботи в мережах PROFINET.

В даний час найбільш яскраво простежуються два напрямки використання мереж PROFINET:

– побудова систем розподіленого вводу / виводу (PROFINET IO);

– побудова модульних систем управління з розподіленим інтелектом – PROFINET CBA (Component Based Automation).

На рис. 2.28 подано приклад мережі PROFINET.

Залежно від функціонального призначення, в мережі PROFINET можуть використовуватися різні механізми обміну даними, різний склад апаратури і інструментальні засоби проєктування.

У системах PROFINET IO прилади польового рівня підключаються безпосередньо до мережі Industrial Ethernet та обслуговуються PROFINET контролером вводу / виводу. Швидкісний обмін даними має циклічний характер і виконується на швидкості 100 Мбіт/с.

Залежно від складу компонентів, що використовуються в такій мережі, забезпечується підтримка обміну даними в реальному масштабі часу (Real Time, RT) і використання тактової синхронізації (Isochronous RT, IRT). При цьому, як активні мережні компоненти для підтримки RT режиму, можуть застосовуватися

комутатори сімейств SCALANCE X100/200/X300/X400, для підтримки IRT режиму – тільки комутатори сімейства SCALANCE X200IRT/XF200IRT. Підтримується можливість інтеграції існуючих мереж Profibus DP до системи PROFINET IO. При цьому головний пристрій підключається до мережі PRO.

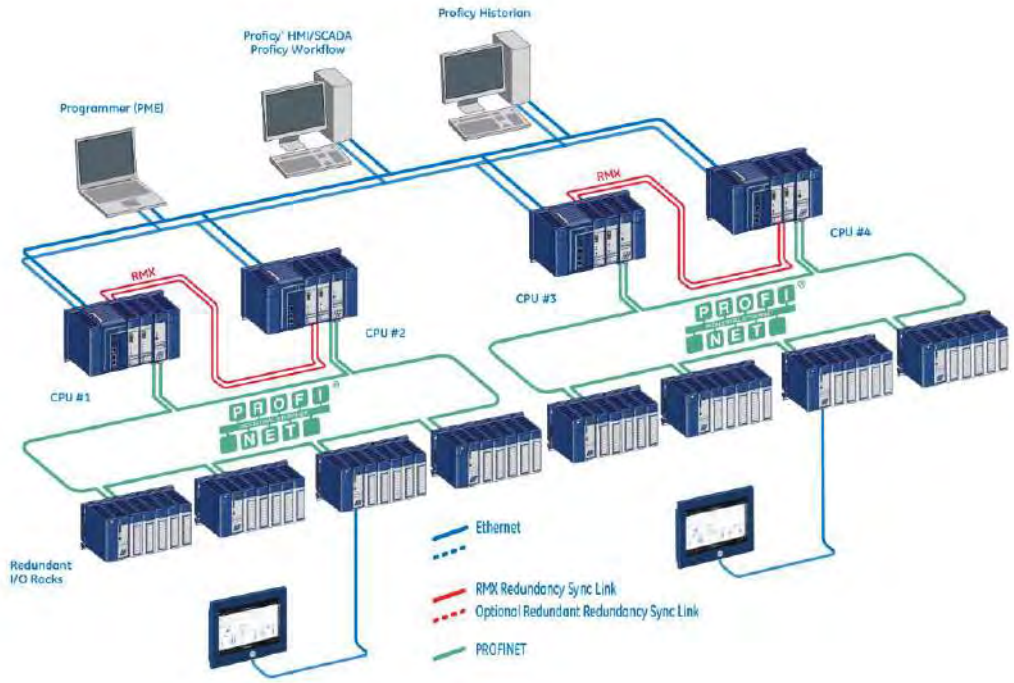


Рисунок 2.28 – Мережа PROFINET

2.4 Контрольні питання

1. Яке призначення моделі взаємодії відкритих систем OSI у промислових мережах?
2. Які функції виконує кожен з 7 рівнів моделі OSI?
3. Що таке передача даних у контексті промислових мереж?
4. Які існують основні способи передачі даних?
5. У чому полягають особливості асинхронної послідовної передачі даних?
6. Як реалізується синхронна послідовна передача даних?
7. Які відмінності між синхронною та асинхронною передачею даних?
8. Яка роль мережної архітектури у багаторівневих системах управління?
9. Які переваги та обмеження має протокол Modbus?

10. Як функціонує шина CANbus і де вона застосовується в автоматизації?

11. У чому полягає суть роботи протоколу Profibus?

12. Які особливості взаємодії пристроїв на польовому рівні?

13. Що таке Actuator Sensor Interface (AS-i) і де він застосовується?

14. Які функції виконує протокол HART у роботі з інтелектуальними вимірювальними приладами?

15. У чому полягають переваги Industrial Ethernet та відкритого стандарту PROFINET у промислових мережах?

16. Які переваги мають послідовний та паралельний способи передачі даних?

17. Для чого використовують універсальний асинхронний приймач / передавач?

18. З яких рівнів складається мережна архітектура багаторівневої системи управління?

19. Які основні властивості має послідовна шина з децентралізованим доступом?

20. Поясніть принцип організації обміну інформацією за протоколом Modbus.

21. Що таке «Польова шина»?

22. Як відбувається передача даних в пакетному режимі з використанням протоколу HART?

3 ПРОМИСЛОВІ ІНТЕРФЕЙСИ

3.1 Інтерфейс RS-232

Інтерфейс RS-232 забезпечує з'єднання двох пристроїв, один з яких називається DTE (Data Terminal Equipment), або КУД (кінцеве устаткування даних) і другий – DCE (Data Communications Equipment), або ОПД (обладнання передачі даних). На рис. 3.1 подано принцип взаємодії між пристроями DTE та DCE.

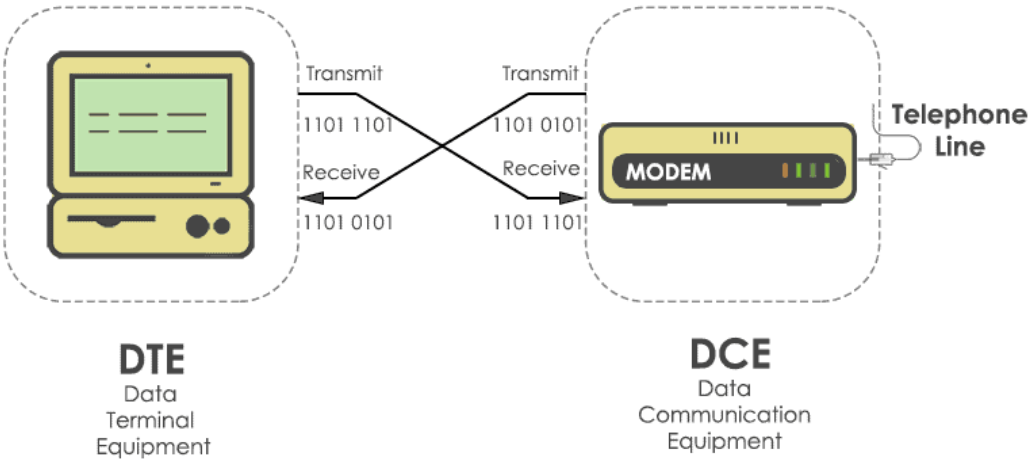


Рисунок 3.1 – Принцип взаємодії між пристроями DTE та DCE

DTE (комп'ютер) послідовно передає інформацію на інше кінцеве обладнання DCE (модем). У прикладі вище, DTE надсилає двійкові дані «11011101» до DCE, а DCE надсилає двійкові дані «11010101» до пристрою DTE.

Стандарт RS-232 описує загальні рівні напруги, електричні стандарти, режим роботи та кількість бітів, що передаються від DTE до DCE. Цей стандарт використовується для передачі обміну інформацією телефонними лініями.

Стандарт визначає максимальну напругу холостого ходу 25 В: зазвичай спостерігаються рівні сигналу ±5 В, ±10 В, ±12 В і ±15 В залежно від напруги, доступної для схеми лінійного драйвера. Деякі мікросхеми драйверів RS-232 мають вбудовану схему для отримання необхідної напруги від джерела живлення 3 В або 5 В. Драйвери та приймачі RS-232 повинні бути здатні витримувати невизначені короткі замикання на землю або будь-який рівень напруги до ± 25 В.

Швидкість наростання або швидкість зміни сигналу між рівнями також контролюється (рис. 3.2).

Signal Voltage Levels	Logical State
-3 to -25	OFF (0)
+3 to +25	ON(1)

Control Signal Voltage Levels (Volts)	Logical State
-3 to -25	OFF (1)
+3 to +25	ON (0)

Рисунок 3.2 – Рівні напруги сигналу в лінії

Напруга сигналу від +3 В до +15 В представляє логічну «1», а напруга сигналу від -3 В до -15 В представляє логічний «0». У той час як сигнали керуючої напруги використовують негативну логіку, тобто логічний «1» вказує від -3 до -15 В, а логічний «0» вказує на +3 В до +15 В (рис. 3.3). Напруга від -3 В до +3 В вважається невизначеним станом.

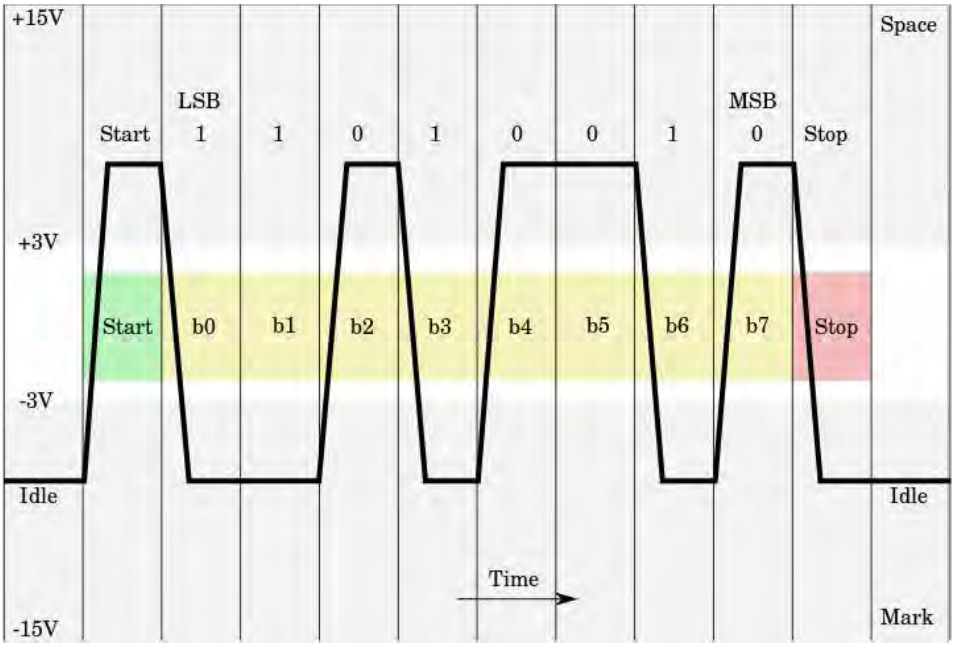


Рисунок 3.3 – Рівні напруги інтерфейсу RS-232

Оскільки рівні напруги вищі за логічні рівні, які зазвичай використовуються інтегральними схемами, для трансляції логічних рівнів потрібні спеціальні проміжні схеми драйверів. Вони також захищають внутрішні схеми пристрою від коротких замикань або перехідних процесів, які можуть

з’явитися на інтерфейсі RS-232, і забезпечують достатній струм для відповідності вимогам до швидкості наростання для передачі даних.

Оскільки обидва кінці ланцюга RS-232 залежать від напруги на контакті заземлення, що дорівнює нулю вольт, виникнуть проблеми під час підключення обладнання та комп’ютерів, де напруга між контактом заземлення на одному кінці та контактом заземлення на іншому не дорівнює нулю.

Це також може спричинити небезпечний контур заземлення. Використання загального заземлення обмежує RS-232 додатками з відносно короткими кабелями. Якщо два пристрої знаходяться на достатній відстані один від одного або в окремих системах живлення, місцеві з’єднання заземлення на обох кінцях кабелю матимуть різну напругу; ця різниця зменшить запас шуму сигналів. Збалансовані диференціальні послідовні з’єднання, такі як RS-422 або RS-485, можуть витримувати більшу різницю напруги на землі завдяки диференціальній сигналізації.

Стандарт RS-232 визначає 25-контактний роз’єм SUB-D як основний. Крім того, 9-контактний SUB-D знайшов широке застосування, особливо в світі ПК.

В таблиці 3.1 подані основні сигнали в роз’ємах та їх призначення.

Таблиця 3.1 – Основні сигнали в роз’ємах DB9 та DB25 інтерфейсу RS-232

DB9	DB25	Позначення	Назва	Опис сигналу
1	8	CD	Carrier Detect	Виявлення несучої
2	3	RXD	Receive Data	Приймання даних
3	2	TXD	Transmit Data	Передача даних
4	20	DTR	Data Terminal Ready	Готовність кінцевого обладнання
5	7	GND	System Ground	Загальний провід
6	6	DSR	Data Set Ready	Готовність обладнання передачі
7	4	RTS	Request to Send	Запит на передачу
8	5	CTS	Clear to Send	Готовий передавати
9	22	RI	Ring Indicator	Наявність сигналу виклику

Розташування контактів в роз’ємі DB9-M подано на рис. 3.4.

Виходячи з таблиці 3.1, контакти мають таке призначення:

– контакт 1: CD (Carrier Detect) виявлення несучої є вхідним сигналом від DCE;

– контакт 2: RD (Receive Data) отримує вхідні дані від DTE;

– контакт 3: TD (Transmit Data) надіслати вихідні дані до DCE;

– контакт 4: DTR (Data Terminal Ready) вихідний сигнал встановлення зв’язку;

– контакт 5: GND (Signal ground) загальна опорна напруга;

– контакт 6: DSR (Data Set Ready) вхідний сигнал встановлення зв’язку;

– контакт 7: RTS (Request to Send) вихідний сигнал для контролю потоку;

– контакт 8: CTS (Clear to Send) вхідний сигнал для керування потоком;

– контакт 9: RI (Ring Indicator) вхідний сигнал від DCE.

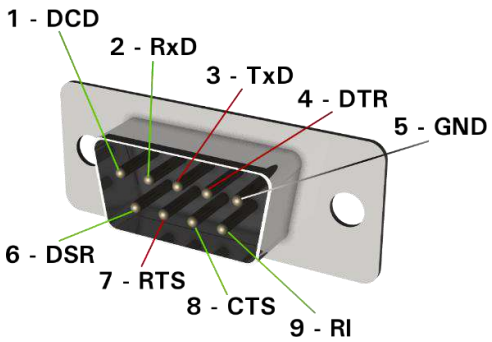


Рисунок 3.4 – Розташування контактів в роз’ємі DB9-M

Роботу RS-232 можна розглянути на прикладі протоколу UART, який є протоколом асинхронного зв’язку «точка-точка». Формат даних ініціюється початковим бітом, за яким слідують 7-бітні двійкові дані, біт парності та стоп-біт, які надсилаються один за одним (рис. 3.5).

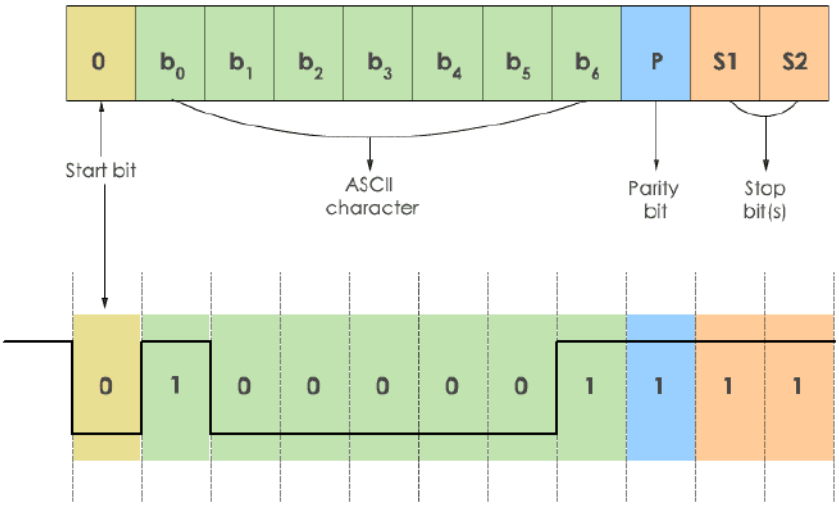


Рисунок 3.5 – Формат кадру протоколу UART

Передача починається з надсилання стартового біта «0». Після цього йдуть 7 біт даних ASCII. Отримавши стартовий біт, приймач вибирає з лінії біти даних через визначені інтервали часу. Дуже важливо, щоб частоти роботи приймача і передавача були однаковими, припустиме розходження – не більше 10%. Швидкість передачі по RS-232 може вибиратися з ряду: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 біт/с.

Біт парності додається до цих даних для перевірки приймача. Дані, надіслані від передавача, повинні збігатися в приймачі. Передача припиняється за допомогою стоп-біта, який представлений двійковою «1». Зазвичай можна надіслати 1 або 2 стоп-біти.

На поданій вище схемі символ ASCII «A» надсилається за допомогою послідовного двійкового потоку «1» і «0». Під час надсилання даних між кожним бітом має бути певна затримка. Ця затримка вважається часом неактивності, а лінія RS232 знаходиться в негативному логічному стані (-12 В).

Принцип визначення стану сигналу в лінії подано на рис. 3.6.

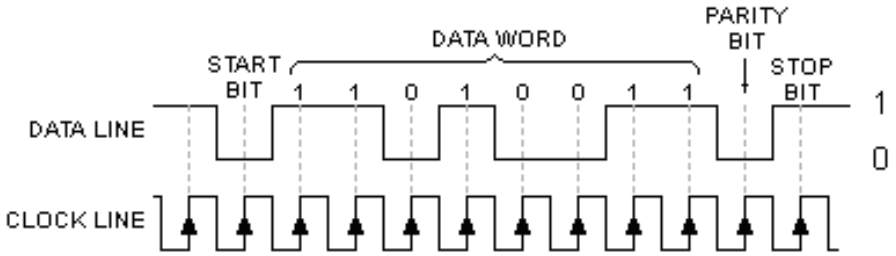


Рисунок 3.6 – Принцип визначення стану сигналу в лінії

Для з’єднання двох приладів через канал RS-232 потрібно використовувати з’єднання ліній передавача з лініями приймача в обох напрямках і загального заземлення. Також потрібно реалізувати апаратне встановлення зв’язку – лінії запиту потрібно з’єднати з лініями готовності в обох напрямках (рис. 3.7).

Якщо апаратна підтримка протоколу не потрібна в каналі зв’язку, нуль-модемний кабель можна побудувати за допомогою лише трьох з’єднувальних проводів і деяких локальних з’єднань для контроля потоку даних (рис. 3.8).

Рисунок 3.7 – Приклад кабелю для поєднання двох пристроїв за допомогою RS-232

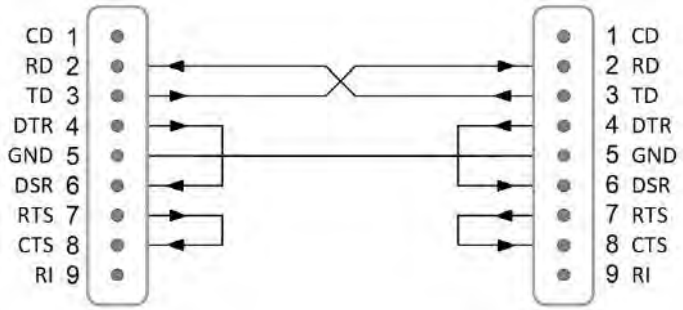


Рисунок 3.8 – Приклад NULL модемного кабелю RS232

Мінімально можливий спосіб поєднання двох пристроїв подано на рис. 3.9.

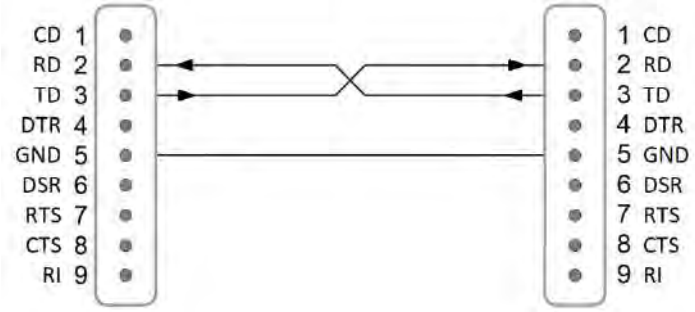


Рисунок 3.9 – Мінімально можливий спосіб поєднання двох пристроїв

Довжина кабелю впливає на максимальну швидкість передачі інформації. Більш довгий кабель має більшу ємність і, відповідно, для забезпечення надійної передачі, більш низьку швидкість. Більша ємність призводить до того, що зміна напруги одного сигнального проводу може передаватися на інший межовий сигнальний провід. На високій швидкості максимальна відстань зазвичай вважається рівною 15 м.

Таблиця 3.2 – Максимальна довжина кабелю для обраної швидкості

Швидкість, біт/с	Максимальна довжина лінії зв'язку, м
19 200	15
9 600	150
4 800	300
2 400	900

Дані можуть передаватись за протоколом керування потоком. Апаратний протокол управління потоком RTS/CTS (апаратний контроль потоку) використовує сигнал CTS, який дозволяє встановити передачу даних, якщо приймач не готовий до їх прийому (рис. 3.10).

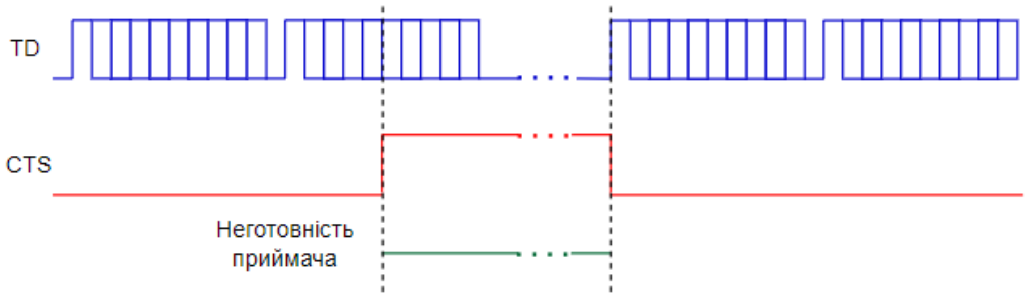


Рисунок 3.10 – Апаратний контроль потоку

Передавач «випускає» черговий байт, якщо включена лінія CTS. Байт, який почав передаватися, затримати сигналом CTS неможливо (це гарантує цілісність посилки).

Окрім апаратного протоколу керування потоком є програмний протокол керування потоком (рис. 3.11). Працює протокол наступним чином: якщо пристрій, що приймає дані, виявляє причини, з яких воно не може їх приймати далі, воно по зворотному послідовному каналу посилає байт-символ XOFF (13h). Протилежний пристрій, прийнявши цей символ, зупиняє передачу. Коли

пристрій знову стає готовим до прийому даних, він посилає символ XON (11h), прийнявши, який протилежний пристрій відновлює передачу.

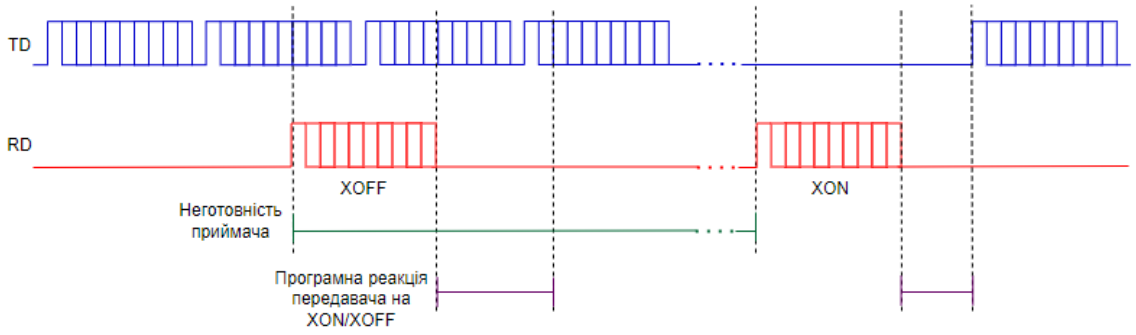


Рисунок 3.11 – Програмний протокол керування потоком

Перевага програмного протоколу полягає у відсутності необхідності передачі керуючих сигналів інтерфейсу – мінімальний кабель для двостороннього обміну може мати лише 3 дроти (див. рис. 3.11). Недоліком, крім обов'язкової наявності буфера і більшого часу реакції (що знижує загальну продуктивність каналу через очікування сигналу XON) є складність реалізації повнодуплексного режиму обміну. У цьому випадку з потоку даних, що приймаються, повинні виділятися (і оброблятися) символи управління потоком, що обмежує набір символів, що передаються.

3.2 Інтерфейси RS-485, RS-482

При використанні інтерфейсу RS-232 виникає серйозна проблема – низька перешкодозахищеність. Сигнали інтерфейсу RS-232 прив'язані до нульового рівня напруги, тому перешкоди з нульового дроту можуть викликати перешкоди, що спотворюють корисний сигнал. Зона нечутливості сигналів інтерфейсу RS-232 становить ± 3 В. Цього може виявитися недостатньо для захисту від миттєвих перепадів напруги цифровими лініями. Якщо схема працює зі стандартними рівнями сигналів ± 5, що відповідає TTL-логіці, то сигнали тактування цифрових схем можуть викликати появу помилкових сигналів на земляній шині в тому випадку, якщо шини живлення розташовані поруч. Такі сигнали можуть інтерпретуватися програмним забезпеченням, як біти посилання, що призведе до помилок.

Подібні обмеження були усунені у протоколі RS-485, який був розроблений, як удосконалений варіант протоколу RS-232. Сигнали інтерфейсу RS-485 не прив'язані до нульового рівня, а є диференціальними, що значно послаблює синфазні перешкоди. Інтерфейс RS-485 дуже широко застосовується для створення мереж промислової автоматики та для надійної передачі між віддаленими об'єктами. Протокол RS-485 є основою популярних промислових мереж управління Profibus і Modbus.

Мережа, побудована на інтерфейсі RS-485, є приймачами, з'єднаними за допомогою крученої пари. В основі інтерфейсу RS-485 є принцип диференціальної (балансної) передачі даних. Суть його полягає в передачі одного сигналу двома дротами. Причому з одного дроту (умовно А) йде оригінальний сигнал, а з іншого (умовно В) – його інверсна копія. Іншими словами, якщо на одному дроті «1», то на іншому – «0», і навпаки. Таким чином, між двома проводами крученої пари завжди є різниця потенціалів: якщо «1» – вона позитивна, якщо «0» – негативна.

Принцип передачі сигналів за інтерфейсом RS-485 подано на рис. 3.12.

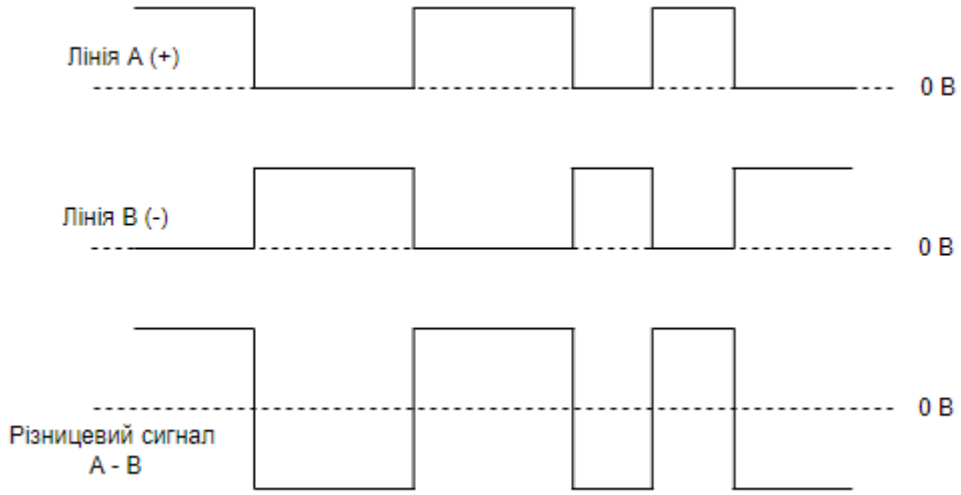


Рисунок 3.12 – Принцип передачі сигналів за інтерфейсом RS-485

Сигнали в лініях А і В можуть набувати невід'ємних значень і є інверсними по відношенню один до одного, а диференціальний сигнал є їх різницею, тобто рівень напруги сигналу в лінії А віднімається з рівня напруги в лінії В. Таким чином, виходить чистий диференціальний сигнал, який не залежить від рівнів

синфазних складових в лініях A і B, оскільки при відніманні різниця таких постійних сигналів дає 0.

Такий метод формування сигналу відрізняється від того, що використовується в RS-232 і базується на рівні напруги щодо загального дроту схеми. Будь-яка перешкода з досить високим рівнем напруги із загального дроту призведе до спотворення сигналу в інтерфейсі RS-232.

Використання диференціальних сигналів в інтерфейсі RS-485 дозволяє здійснювати взаємодію між вузлами, що знаходяться на значних відстанях один від одного (до 1200 м), що набагато більше, ніж дозволяє протокол RS-232 (до 15 м). Максимальна швидкість обміну даними протоколом RS-485 дорівнює 10 Мбіт/с, хоча сучасні мікросхеми драйверів інтерфейсу RS-485 дозволяють працювати зі швидкостями до 35 Мбіт/с.

Протокол RS-485 дозволяє будувати розгалужені мережі передачі даних за високої надійності, що робить його дуже популярним під час створення мереж промислової та лабораторної автоматики (рис. 3.13). Це єдиний протокол, що дозволяє здійснювати взаємодію між кількома передавачами та приймачами, що знаходяться в тій самій мережі. Якщо використовувати стандартні приймачі RS-485 з вхідним опором 12 кОм, можна створити мережу, що міститиме 32 пристроїв. В даний час випускаються драйвери інтерфейсу RS-485 з вхідним опором до 100 кОм та вище, що дозволяє підключати до однієї мережі до 256 пристроїв.

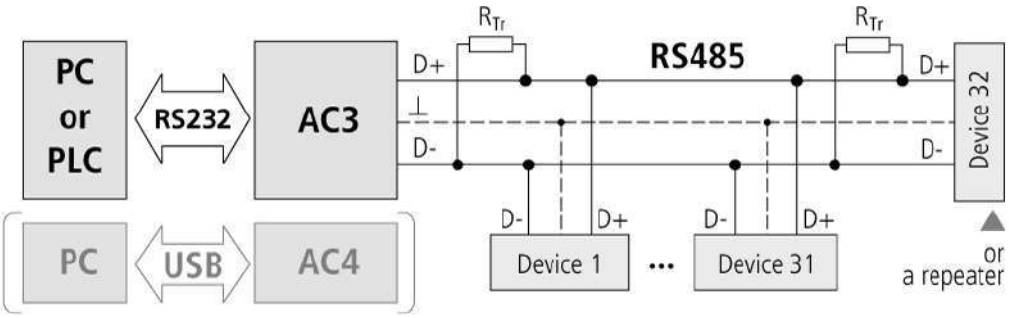


Рисунок 3.13 – Мережа на основі інтерфейсу RS-485

Розширити мережі на базі протоколу RS-485 можна також за рахунок застосування підсилювачів (репітерів) сигналів, що дасть можливість включати до мережі кілька сотень вузлів та збільшити відстань, на якій вони взаємодіють, до кількох кілометрів. Істотною перевагою інтерфейсу RS-485 є простота його

реалізації, порівнянна з протоколом RS-232. З цих причин інтерфейс RS-485 набув дуже широкої популярності для створення мереж на базі ПК, мікроконтролерів та інтелектуальних вимірювальних систем.

Топологію мережі на основі протоколу RS-485 подано на рис. 3.14.

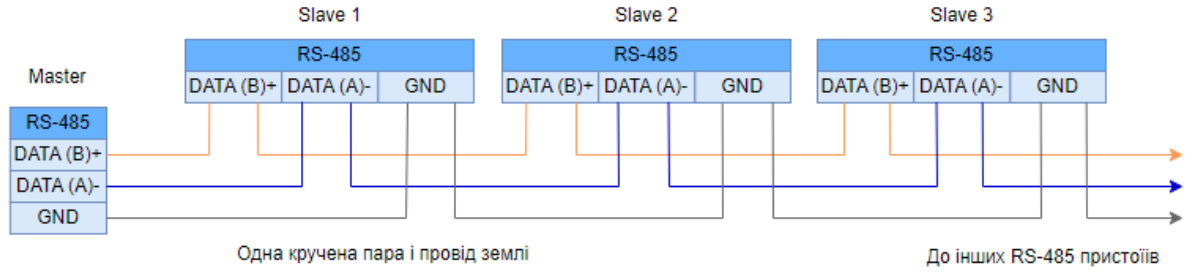


Рисунок 3.14 – Топологія мережі на основі протоколу RS-485

На цьому рисунку подано мережу на базі RS-485, що складається з N вузлів. Для забезпечення високої швидкості передачі даних та зниження рівня відбитих сигналів у довгих лініях на обох кінцях лінії потрібно встановлювати термінуючі резистори, опір яких може перебувати в межах 100-120 Ом. Топологія мережі розроблена за принципом «загальної шини». Реалізація мережі у вигляді «зірки» не допускається через неможливість правильного термінування ліній, що призводить до суттєвого падіння продуктивності. Сигнальна лінія повинна виконуватися «крученою парою», до якої приєднуються всі пристрої шини. При цьому відстань від лінії до мікросхем інтерфейсу RS-485 повинна бути мінімальною.

На рис. 3.15 подана правильна (рис. 3.15, а) та неправильна (рис. 3.15, б) топологія мережі на основі інтерфейсу RS-485. Квадратиками позначені пристрої з інтерфейсом RS-485.

Топологія мереж на основі інтерфейсу RS-485 визначається необхідністю запобігти відбиттю сигналів в лінії зв’язку. Оскільки відбиття виникають через неоднорідності в лінії зв’язку, а також через відгалуження від неї, то єдиною правильною топологією мережі буде така, що виглядає як єдина лінія без відведення, до якої не більше ніж у 32 точках під'єднані пристрої з інтерфейсом RS-485 (рис. 3.15, а, б). Будь-які варіанти, в яких лінія має довгі відведення або з'єднання декількох кабелів в одній точці (рис. 3.15, в-д), призводять до виникнення відбиття та зниження якості передачі.

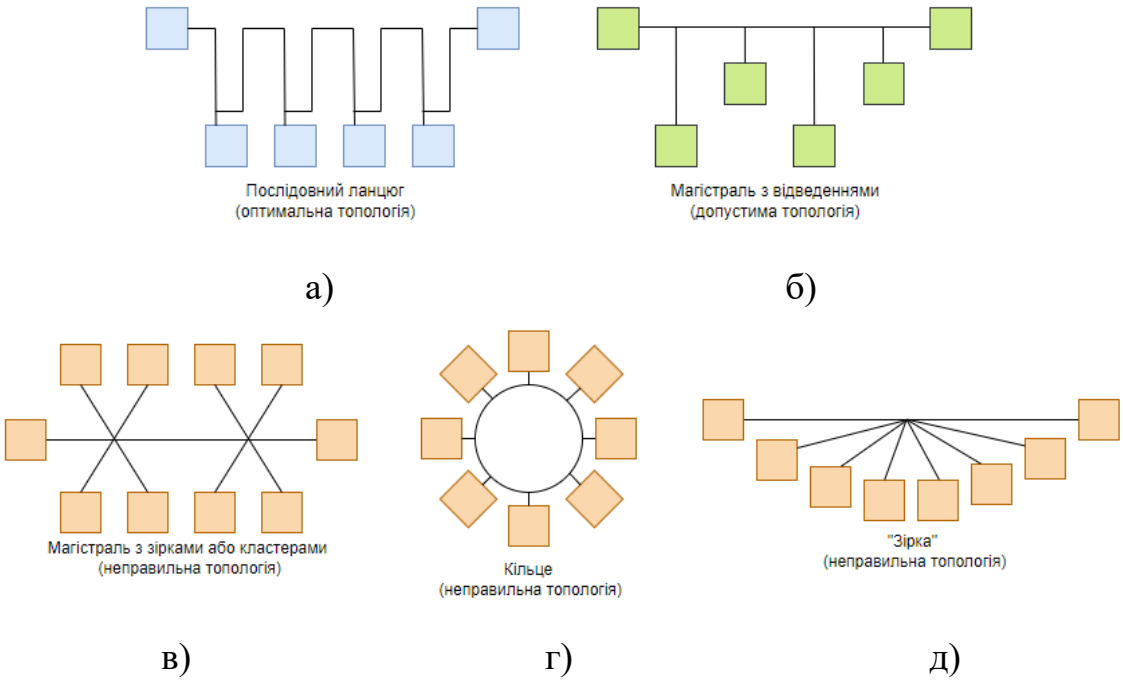


Рисунок 3.15 – Оптимальна (а), допустима (б) та неправильні (в-д) топології мережі на основі інтерфейсу RS-485

Проте, сказане справедливо лише для високої швидкості передачі (вище 9600 біт/с), коли ефекти відбиття впливають на якість передачі сигналу. Для низьких швидкостей довжина відведення може бути довільною.

Якщо є необхідність розгалуження лінії, це можна зробити за допомогою повторювачів інтерфейсу (рис. 3.17) чи концентратора (хаба). Повторювачі дозволяють розділити лінію на сегменти, у кожному з яких виконуються умови узгодження за допомогою двох термінальних резисторів і не виникають ефекти, пов'язані з відбиттям від кінців лінії, а довжина відведення від лінії до повторювача завжди може бути досить малою.

Інтерфейс RS-485, на відміну від RS-232, працює у напівдуплексному режимі. Приймання та передавання відбувається однією парою проводів із розділенням за часом. У мережі може бути багато передавачів, які вимикаються в режимі приймання.

За замовчуванням всі передавачі на шині RS-485 знаходяться у високоімпедансному стані. За допомогою програмного забезпечення вищого рівня такої мережі один із вузлів шини призначається «провідним»: він керує запитами та ініціює команди. Всі інші вузли працюють у режимі «підлеглого» та отримують запити та команди «майстра». Взаємодія з «провідним» одночасно

може здійснювати кілька вузлів, при цьому синхронізація такої взаємодії виконується певними рівнями програмного забезпечення конкретної мережі. Протокол RS-485 є основою таких популярних мережних протоколів, як Profibus і Modbus, які дуже широко використовують у промисловості.

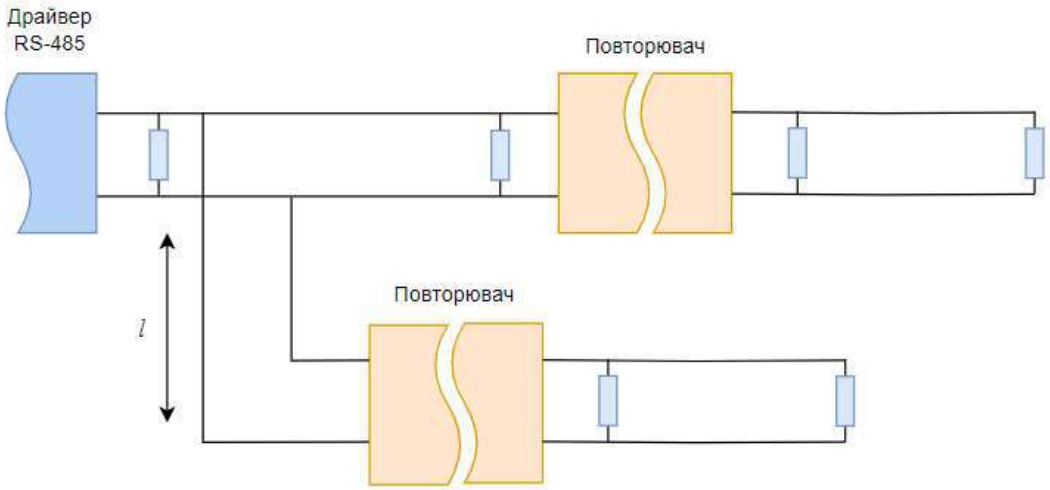


Рисунок 3.17 – Застосування повторювачів інтерфейсу для розгалуження лінії передачі

Інтерфейс RS-485 має дві версії: двопровідну та чотирипровідну. Двопровідна використовується для напівдуплексної передачі (рис. 3.14), коли інформація може передаватися в обох напрямках, але у різний час.

Для повнодуплексної (дуплексної) передачі використовують чотири лінії зв'язку: за двома інформація передається в одному напрямку, за двома іншими – у зворотному (рис. 3.18).

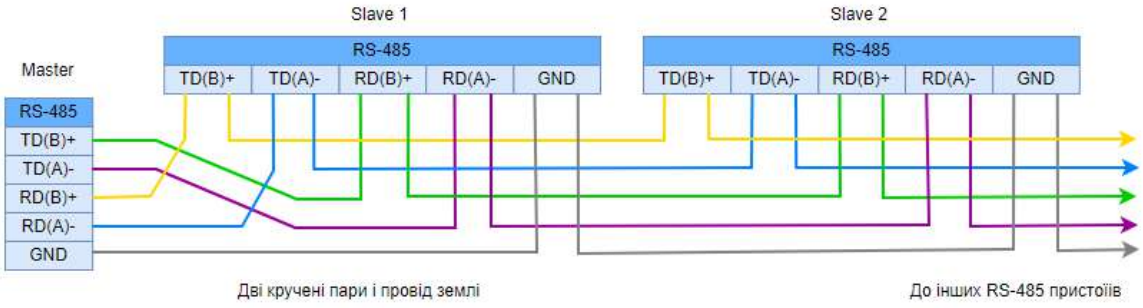


Рисунок 3.18 – Чотирипровідне з'єднання пристроїв з інтерфейсом RS-485

Недоліком чотирипровідної схеми є необхідність у жорсткому визначені головного та підлеглих пристроїв на стадії проєктування системи, у той час як у двопровідній схемі будь-який пристрій може бути як у ролі головного, так і у

ролі підлеглого. Перевагою чотирипровідної схеми є можливість одночасного передавання та приймання даних, що буває необхідно в процесі реалізації деяких складних протоколів обміну.

Коли передавачі всіх пристроїв, підключених до лінії, знаходяться у третьому (високоомному) стані, логічний стан лінії та входів усіх приймачів не визначено. Щоб усунути цю невизначеність, не інвертуючий вхід приймача з'єднують через резистор з шиною живлення, а інвертуючий – з шиною «землі». Величини резисторів вибирають такими, щоб напруга між входами стала більшою за поріг спрацьовування приймача (+ 200 мВ).

Оскільки ці резистори виявляються підключеними паралельно лінії передачі, то для забезпечення узгодження лінії з інтерфейсом необхідно, щоб еквівалентний опір на вході лінії дорівнював 120 Ом.

Наприклад, якщо резистори, що використовуються для усунення невизначеності стану лінії, мають опір 450 Ом кожне, то резистор для узгодження лінії повинен мати номінал 130 Ом, тоді еквівалентний опір ланцюга дорівнює 120 Ом. Для того, щоб знайти диференціальну напругу лінії в третьому стані всіх передавачів (рис. 3.19), потрібно врахувати, що до протилежного кінця лінії стандартної конфігурації підключений ще один резистор опором 120 Ом і до 32 приймачів з вхідним диференціальним опором 12 кОм. Тоді, якщо напруга живлення 5 В, диференціальна напруга лінії дорівнюватиме + 272 мВ, що задовольняє вимогам стандарту.

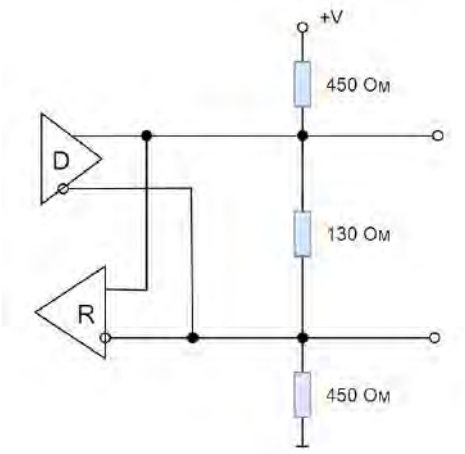


Рисунок 3.19 – Резистивний ланцюг на виході трансивера інтерфейсу, що усуває невизначений стан лінії та забезпечує її узгодження

Інтерфейс RS-422 використовується набагато рідше, ніж RS-485 і, як правило, не для створення мережі, а для з'єднання двох пристроїв на великій відстані (до 1200 м), оскільки інтерфейс RS-232 працездатний лише на відстані до 15 м.

Швидкість передачі в RS-422 залежить від відстані і може змінюватися в межах від 10 кбіт/с (до 1200 м) до 10 Мбіт/с (до 10 м).

У мережі RS-422 може бути тільки один передавальний пристрій і до 10 пристроїв, що приймають.

Лінія RS-422 являє собою 4 дроти для приймання-передавання даних (2 кручені проводи для передавання і 2 кручені проводи для приймання) і один загальний провід землі GND (рис. 3.20).

Скручування проводів (кручена пара) між собою дозволяє позбутися наведень і перешкод, тому що наведення однаково діє на обидва проводи, а інформація витягується з різниці потенціалів між провідниками A і B одній лінії.

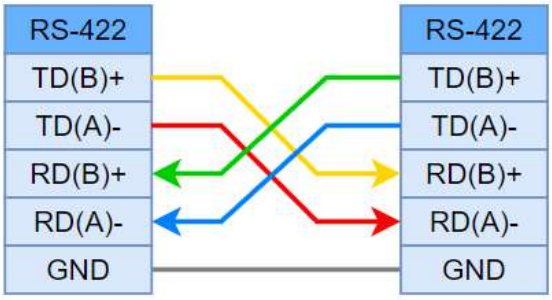


Рисунок 3.20 – Поєднання двох пристроїв за допомогою інтерфейсу RS-422

Відстань між приймачем і передавачем RS-422 може досягати 1200 м, тому для запобігання відбиття сигналу від кінця лінії ставиться спеціальний узгоджуючий резистор 120 Ом. Цей резистор встановлюється між RX+ та RX- контактами на початку та в кінці лінії.

Порівняння інтерфейсів RS-232, RS-422 і RS-485 подано в таблиці 3.3.

На комп'ютері інтерфейси RS-232/422/485 будуть представлені як звичайний порт COM. Відповідно можна використовувати майже будь-які програми та утиліти для роботи з COM портом.

Кожен виробник випускає своє програмне забезпечення для роботи з COM портом. Наприклад, на рис. 3.21 подано інтерфейс налаштування параметрів інтерфейсу для організації обміну даними з пристроєм через COM-порт.

Таблиця 3.3 – Порівняння інтерфейсів RS-232, RS-422 і RS-485

Параметр	RS-232	RS-422	RS-485
Спосіб передачі сигналу	Однофазний	Диференціальний	Диференціальний
Максимальна кількість приймачів	1	10	32
Максимальна довжина кабелю	15 м	1200 м	1200 м
Максимальна швидкість передачі	460 кбіт/с	10 Мбіт/с	30 Мбіт/с
Синфазна напруга на виході	± 25 В	-0,25...+6 В	-7...+12 В
Напруга в лінії під навантаженням	± 5... ± 15 В	± 2 В	± 1,5 В
Імпеданс навантаження	3...7 кОм	100 Ом	54 Ом
Струм утікання в «третьому» стані	-	-	± 100 мкА
Припустимий діапазон сигналів на вході приймача	± 15 В	± 10 В	-7...+12 В
Чутливість приймача	± 3 В	± 200 мВ	± 200 мВ
Вхідний опір приймача	3...7 кОм	4 кОм	12 кОм

Для підключення до мережі RS-485 в даному випадку використовується інтерфейсний модуль. Модуль конвертера UART-RS485 побудований на мікросхемі MAX485 і призначений для підключення пристроїв з поширеним інтерфейсом RS485, що є промисловим стандартом підключення периферійних пристроїв. Малі розміри модуля дозволяють створювати компактні пристрої.

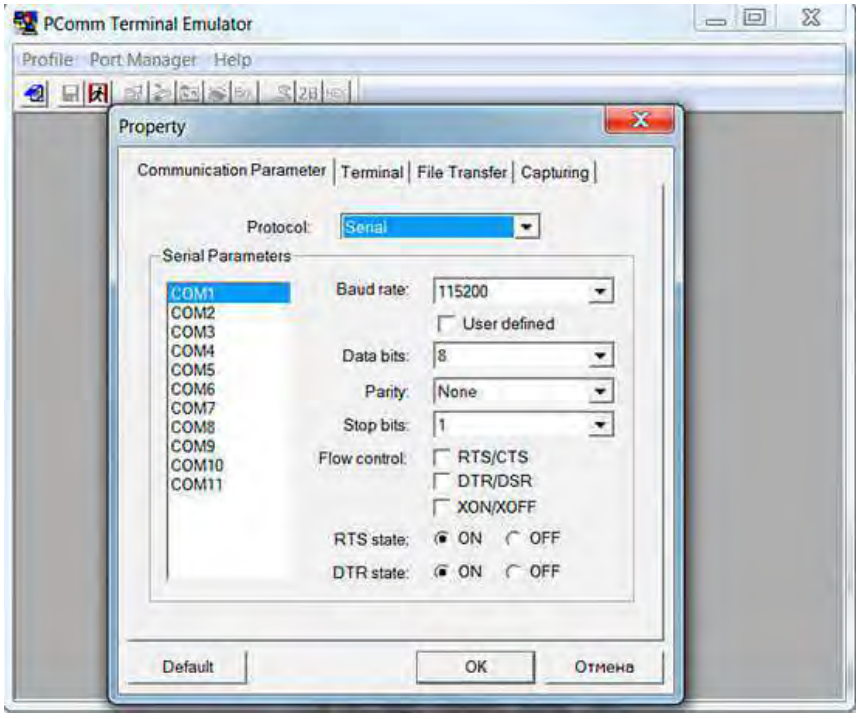


Рисунок 3.21 – Інтерфейс налаштування параметрів інтерфейсу для організації обміну даними з пристроєм через COM-порт

Характеристики пристрою:

– мікросхема перетворювача MAX485;

– вхідний інтерфейс UART;

– вихідний інтерфейс RS-485;

– напруга живлення 5В;

– розміри модуля 44 мм × 14 мм.

Наприклад інтерфейсний модуль можна використовувати, як на боці Raspberry PI, так і на боці модуля керування рухом. В кожному випадку використовується той самий тип модуля, що подано на рис. 3.22.

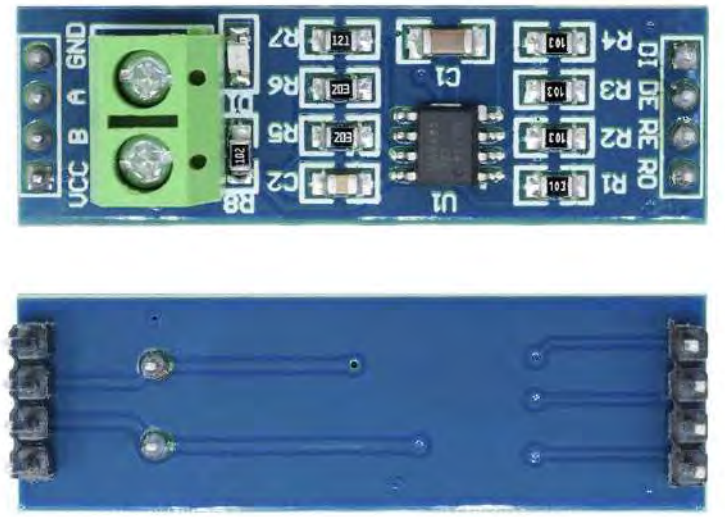


Рисунок 3.22 – Інтерфейсний модуль MAX485 UART-RS485

Для підключення до мережі використовуються клеми «A» та «B». Контакти «DI», «DR», «RE», «RO» використовуються для підключення до мікроконтролера.

На рис. 3.23 подано приклад підключення модулю до Raspberry PI. Підключення провідників наступне:

– Raspberry PI (UART_TXD) – RS-485 (RO);

– Raspberry PI (UART_RXD) – RS-485 (DI);

– Raspberry PI (GPIO 17) – RS-485 (RE);

– Raspberry PI (GPIO 27) – RS-485 (DE).

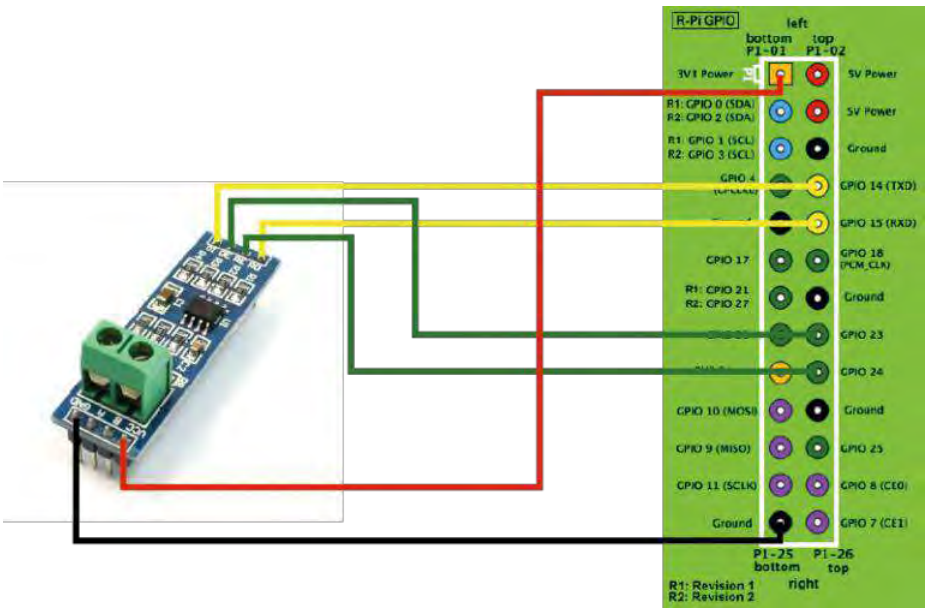


Рисунок 3.23 – Приклад підключення модулю до Raspberry PI

На рис. 3.24 подано приклад організації мережі із трьох пристроїв, кожний з них підключається до загальної шини за допомогою інтерфейсного модуля.

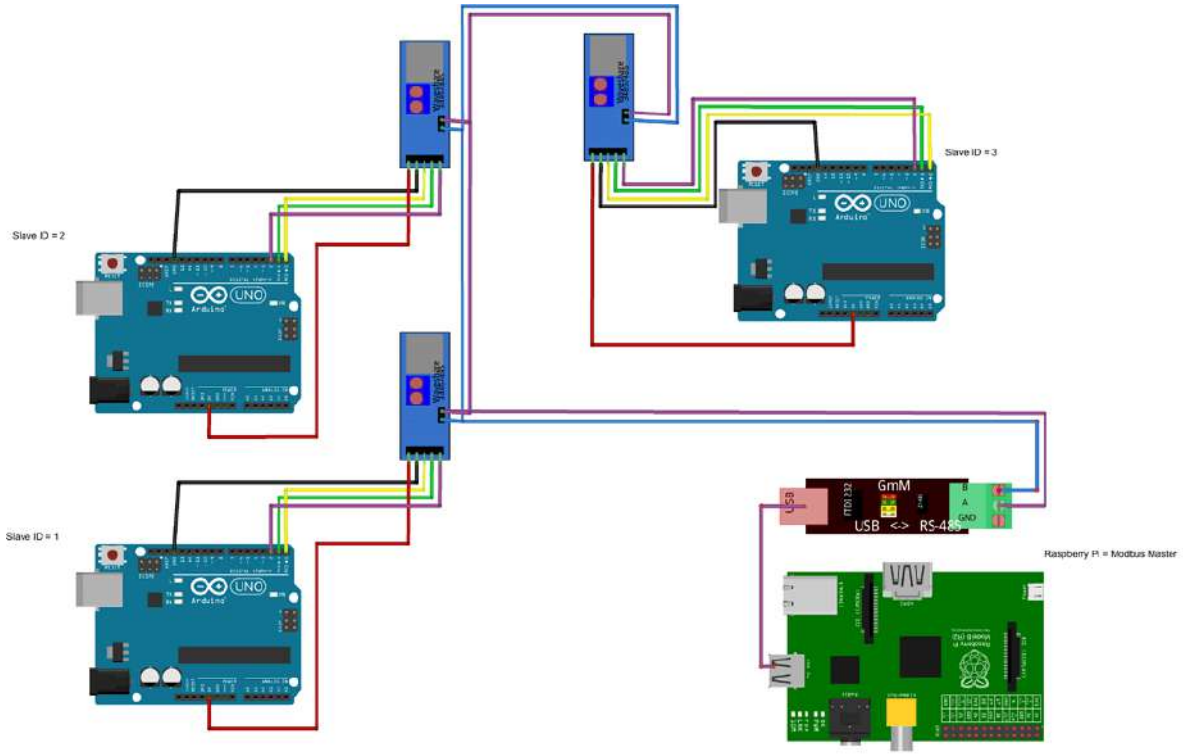


Рисунок 3.24 – Приклад організації мережі із трьох пристроїв

3.3 Контрольні питання

1. Яке призначення інтерфейсу RS-232 у промислових системах автоматизації?
2. Які основні характеристики та параметри інтерфейсу RS-232?
3. Скільки пристроїв може бути підключено до одного порту RS-232?
4. Які типи сигналів використовуються в інтерфейсі RS-232?
5. У чому полягають обмеження відстані та швидкості передачі RS-232?
6. Які типи з’єднувачів використовуються для реалізації RS-232?
7. Що таке повнодуплексний режим передачі даних і чи підтримується він у RS-232?
8. Яка різниця між інтерфейсами RS-232 і RS-485?
9. Які переваги RS-485 у порівнянні з RS-232 для промислового застосування?
10. Скільки пристроїв може бути підключено до однієї шини RS-485?
11. Які способи підключення (топологія) підтримує інтерфейс RS-485?
12. Що таке диференціальна передача сигналів і як вона реалізована в RS-485?
13. У чому полягає принцип роботи інтерфейсу RS-422 (RS-482) і чим він відрізняється від RS-485?
14. У яких випадках доцільно використовувати інтерфейс RS-422 замість RS-485?
15. Які типові сфери застосування інтерфейсів RS-232, RS-485 та RS-422 у промисловій автоматизації?
16. Яка напруга вважається невизначеним станом під час роботи з інтерфейсом RS-232?
17. Назвіть основні сигнали в роз’ємах DB9 та DB25 інтерфейсу RS-232.
18. Назвіть недоліки та переваги повнодуплексної (дуплексної) передачі даних інтерфейсу RS-485.
19. Які характеристики має інтерфейс RS-422?

4 ПРОТОКОЛ MODBUS

4.1 Базові поняття

Протокол Modbus та мережа Modbus [15] є найпоширенішими у світі. Незважаючи на свій вік (стандартом Modbus став ще 1979 року), Modbus не лише не застарів, але, навпаки, суттєво зросла кількість нових розробок та обсяг організаційної підтримки цього протоколу. Мільйони Modbus-пристроїв у всьому світі продовжують успішно працювати.

Однією з переваг Modbus є відсутність потреби у спеціальних інтерфейсних контролерах (Profibus та CAN вимагають для своєї реалізації замовні мікросхеми), простота програмної реалізації та елегантність принципів функціонування. Все це знижує витрати на освоєння стандарту системними інтеграторами і розробниками контролерного обладнання.

Основним недоліком Modbus є мережний обмін за типом «головний / підлеглий», що не дозволяє підлеглим пристроям передавати дані в міру їх появи і тому потребує інтенсивного опитування головними пристроями.

Різновидами Modbus є протоколи Modbus Plus – багатомайстерний протокол із кільцевою передачею маркера та Modbus TCP, розрахований на використання в мережах Ethernet та інтернет.

Протокол Modbus має два режими передачі: RTU (Remote Terminal Unit – «віддалений термінальний пристрій») та ASCII. Стандарт передбачає, що режим RTU в протоколі Modbus повинен бути обов'язково, а режим ASCII є опціонним. Користувач може вибирати будь-який з них, але всі модулі, включені в мережу Modbus, повинні мати той самий режим передачі.

Modbus RTU допускає один активний (ініціатор) пристрій в лінії (Master), який може передавати команди одному або декільком пасивним пристроям (Slave), звертаючись до них за унікальною в лінії адресою. Синтаксис команд протоколу дозволяє адресувати до 255 пристроїв на одній лінії зв'язку стандарту RS-232 або RS-485.

Ініціатива проведення обміну завжди виходить від активного (Master) пристрою, наприклад ПК. Пасивні пристрої прослуховують лінію зв'язку. Активний пристрій надсилає запит (посилання, послідовність байт) в лінію і переходить в стан прослуховування лінії зв'язку. Пасивний пристрій відповідає

на запит, що прийшов на його адресу. Активний пристрій має встановлювати інтервал часу на приймання посилки даних. Якщо цей інтервал перевищив заданий час то генерується таймаут приймання даних.

Протокол забезпечує взаємодію у режимі Request / Response. Майстер ініціює запит до підлеглого пристрою, передаючи в PDU код функції та дані. Залежно від фізичного рівня мережі в пакеті можуть бути додаткові поля, розглянуті вище. На рис. 4.1 подано принцип роботи протоколу Modbus у випадку відсутності помилок на підлеглому пристрої. Якщо обробка запиту проходить без помилок, підлеглий пристрій повертає пакет, що містить вихідний код функції та запитані дані.

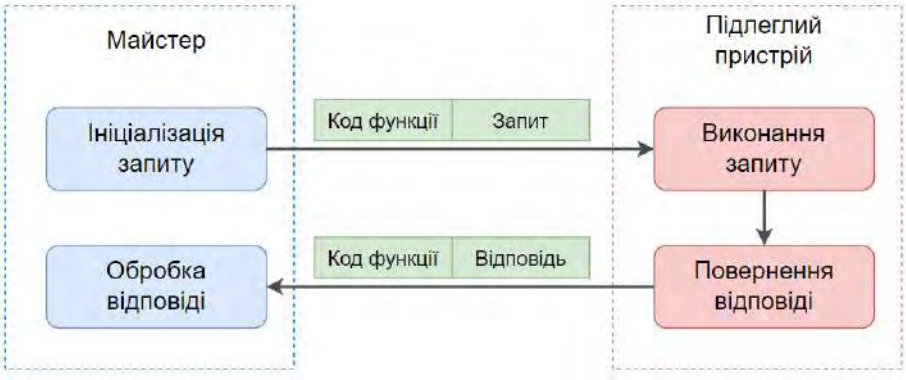


Рисунок 4.1 – Принцип роботи протоколу Modbus у випадку відсутності помилок на підлеглому пристрої

При виникненні помилки підлеглий пристрій повертає в якості даних код помилки, а замість вихідного коду функції – значення цієї помилки, збільшене на 128 (0x80 в шістнадцятковій системі HEX) (рис. 4.2).

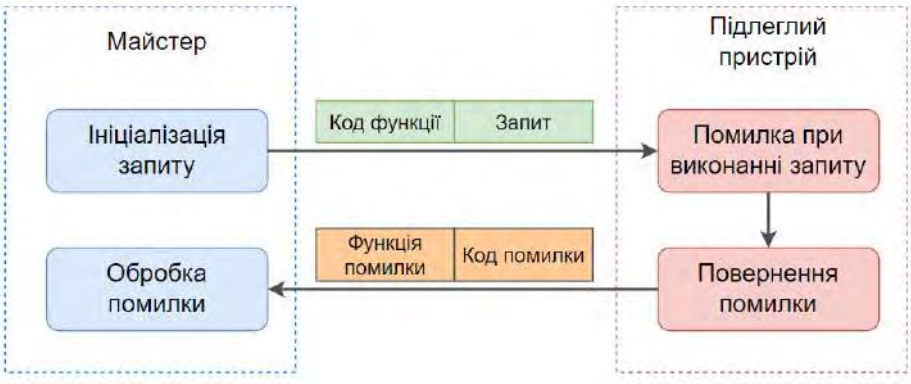


Рисунок 4.2 – Схема роботи Modbus у випадку помилок на підлеглому пристрої

Також передбачені тайм-аути на стороні майстра, щоб уникнути тривалого очікування відповіді від пристроїв, що вийшли з ладу.

4.2 Організація обміну даними за протоколом Modbus

4.2.1 Різновиди протоколу Modbus

Modbus – це протокол прикладного (сьомого) рівня моделі OSI. Він не залежить від нижчих рівнів і може використовуватися спільно з іншими протоколами, наприклад Ethernet TCP/IP або UDP/IP, а як фізичне середовище для передачі сигналів застосовувати послідовні інтерфейси RS-232, RS-422, RS-485, оптоволокно, радіоканали тощо (рис. 4.3).

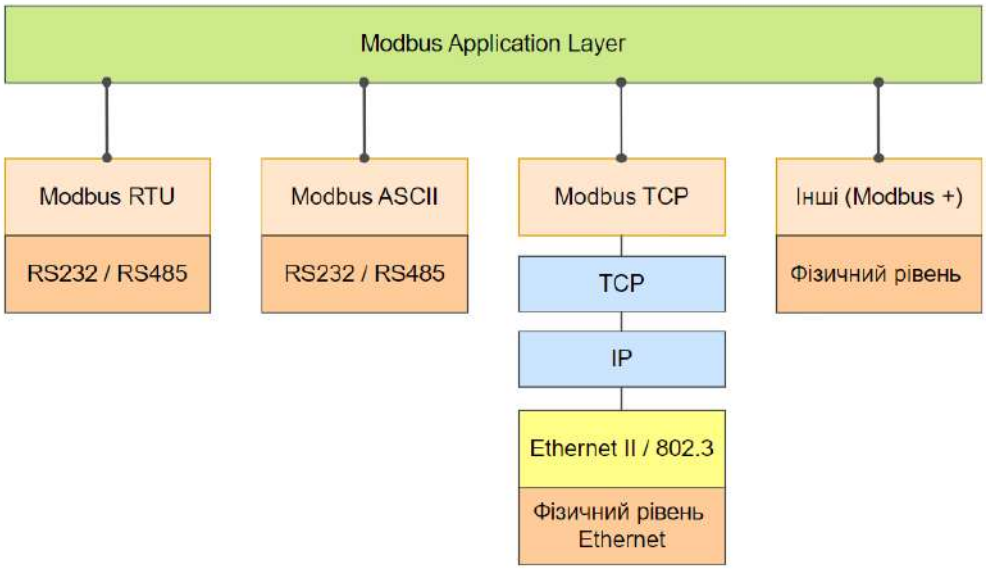


Рисунок 4.3 – Різновиди протоколу Modbus

Протокол Modbus передбачає, що тільки один головний пристрій (контролер) і до 247 підлеглих (модулів вводу / виводу) можуть бути об'єднані в промислову мережу. Обмін даними завжди ініціюється головним. Підлеглі пристрої ніколи не починають передачу даних, доки не отримають запит від головного. Підлеглі пристрої не можуть обмінюватися даними один з одним. Тому будь-якої миті в мережі Modbus може відбуватися лише один акт обміну.

Адреси з 1 до 247 є адресами Modbus пристроїв у мережі, а з 248 до 255 зарезервовані. Головний пристрій не повинен мати адреси та в мережі не повинно бути двох пристроїв з однаковими адресами.

Головний пристрій може надсилати запити всім пристроям одночасно

(широкомовний режим) або тільки одному. Для широкомовного режиму зарезервовано адресу «0» (при використанні команди цієї адреси вона приймається всіма пристроями мережі).

У протоколі Modbus RTU повідомлення починає сприйматися як нове після паузи (тиші) на шині тривалістю щонайменше 3,5 символів (14 біт), тобто тривалість паузи залежить від швидкості передачі. На рис. 4.4 подано формат кадру протоколу Modbus.

ADU – «Application Data Unit» – «елемент даних додатка»
PDU – «Protocol Data Unit» – «елемент даних протоколу»

Рисунок 4.4 – Формат кадру Modbus

Поле адреси завжди містить тільки адресу пристрою, навіть у відповідях на команду, надіслану майстром. Завдяки цьому головний пристрій знає, від якого модуля надійшла відповідь.

Поле «Адреса» (Additional address) визначає, за якою адресою слід надіслати запит від майстра. Може приймати значення від 1 до 247. Адреса 0 використовується для широкомовної передачі даних від майстра всім підлеглим пристроям (відповідь підлеглого пристрою при цьому не передбачена), а адреси 248-255 вважаються зарезервованими. У деяких реалізаціях протоколу поле ігнорується – наприклад, Modbus TCP, де найчастіше застосовується стандартна IP-адресація.

Поле «Код функції» говорить модулю про те, яку дію потрібно виконати. Це поле визначає, яку дію необхідно виконати на підлеглому пристрої. Значення кодів функцій лежать від 1 до 255, причому коди від 128 до 255 зарезервовані для повідомлень про помилки. Код 0 не використовується.

Для кодів з діапазонів 65-72 та 100-110 користувачі можуть реалізувати власні функції (User-Defined Function Codes). Деякі коди, наприклад 9, 10, 13 та інші, зарезервовані певними постачальниками для обладнання та закриті для

загального використання (Reserved Function Codes). Коди, що не входять до цих двох підмножин, відносяться до публічних (Public Function Codes) – це задокументовані функції, що знаходяться у відкритому доступі.

Поле «Дані» може містити довільну кількість байт. У ньому може бути інформація про параметри, що використовуються в запитах контролера або відповіді модуля. Дані, необхідні для виконання вибраної функції підлеглим пристроєм. Найчастіше це адреси регістрів для читання чи запису, їх кількість тощо. Довжина та формат поля залежать від коду функції. Деякі функції не потребують передачі даних.

Поле «Контрольна сума» містить контрольну суму CRC довжиною 2 байти. Дане поле містить розраховане за допомогою спеціального алгоритму число перевірки цілісності пакета. Як алгоритм для розрахунків використовується CRC-16 або LRC-8. У деяких реалізаціях протоколу поле відсутнє – наприклад, Modbus TCP, де контроль цілісності пакета забезпечується засобами протоколу TCP/IP.

4.2.2 Modbus RTU (Remote Terminal Unit)

Modbus RTU (Remote Terminal Unit) – це різновид протоколу, який як фізичний рівень мережі найчастіше використовує послідовний інтерфейс RS-485, рідше RS-232 і RS-422. По суті, всі ці інтерфейси визначають зв'язок за допомогою кручених пар, але відрізняються характеристиками виду максимальної довжини кабелю, кількості вузлів і таке інше.

Формат пакета Modbus RTU загалом збігається з узагальненою формою, описаною раніше: додаткові поля не використовуються. Контроль цілісності пакетів здійснюється за допомогою алгоритму CRC-16.

За замовчуванням в RTU режимі біт паритету встановлюють рівним 1, якщо кількість двійкових одиниць у байті непарне, і рівним 0, якщо воно парне. Такий паритет називають парним (even parity) та метод контролю називають контролем парності (рис. 4.5).

Якщо біт паритету відсутній, на його місце записується другий стоп-біт.

У випадку, якщо кількість двійкових одиниць у байті парний, біт паритету може дорівнювати 1. У цьому випадку кажуть, що паритет є непарним (odd parity).

Контроль парності може бути відсутнім взагалі. У цьому випадку замість біту паритету має використовуватися другий стоповий біт. Для забезпечення

максимальної сумісності з іншими продуктами рекомендується використовувати заміну біта паритету на другий стоповий біт.

Ведені пристрої можуть сприймати будь-який з варіантів: парний, непарний паритет або його відсутність.

Рисунок 4.5 – Послідовність бітів у режимі RTU

Повідомлення Modbus RTU передаються у вигляді кадрів, кожному з яких відомо початок і кінець. Ознакою початку кадру є пауза (тиша) тривалістю щонайменше 3,5 шістнадцяткових символів (14 біт). Кадр повинен передаватися безперервно. Якщо під час передачі кадру виявляється пауза тривалістю більше 1,5 шістнадцяткових символу (6 біт), то вважається, що кадр містить помилку і повинен бути відхилений приймаючим модулем. Ці величини пауз повинні суворо дотримуватися при швидкостях нижче 19200 біт/с, проте за більш високих швидкостях рекомендується використовувати фіксовані паузи, 1,75 мс і 750 мкс відповідно.

Приклад пакету даних в форматі Modbus RTU подано на рис. 4.6.

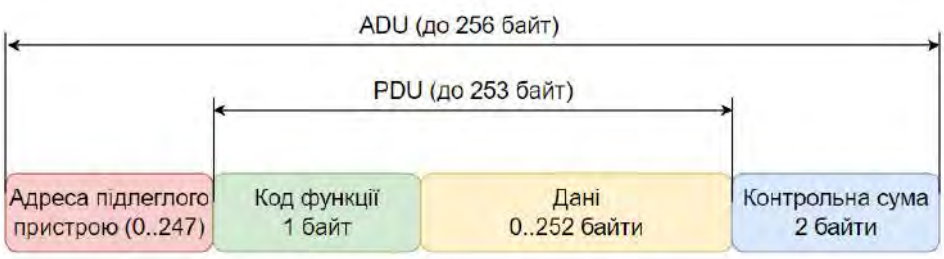


Рисунок 4.6 – Приклад пакету даних в форматі Modbus RTU

У режимі RTU є два рівні контролю помилок у повідомленні:

– контроль паритету для кожного байту (опційно);

– контроль кадру загалом за допомогою CRC методу.

Метод CRC використовується незалежно від перевірки паритету. Значення CRC встановлюється у головному пристрої перед передачею. Під час прийому

повідомлення обчислюється CRC всього повідомлення і порівнюється з його значенням, зазначеним у полі CRC кадру. Якщо обидва значення збігаються, вважається, що повідомлення не містить помилки. Стартові, стопові біти та біт паритету у обчисленні CRC не беруть участі.

4.2.3 Modbus ASCII

Modbus ASCII – це різновид протоколу Modbus, що також працює поверх інтерфейсів RS-232/485, але для кодування повідомлень використовує ASCII-символи. У порівнянні з Modbus RTU у форматі пакета додаються ще два поля – спеціальні символи для позначки початку та кінця повідомлення: двокрапка та символи повернення каретки / перекладу рядка (рис. 4.7). Часові паузи між пакетами не потрібні. Для перевірки цілісності використовується алгоритм LRC-8.

Рисунок 4.7 – Структура пакета даних в Modbus ASCII

Загалом цей варіант протоколу зараз використовується вкрай рідко – через складності кодування та великий розмір повідомлень. Однак він може стати гарною альтернативою Modbus RTU на лініях з мережними затримками та устаткуванні з менш точними таймерами.

4.2.4 Modbus TCP

Modbus TCP – це реалізація ModBus у мережах Ethernet, що працює поверх TCP/IP стека.

На відміну від Modbus RTU і ASCII, Modbus TCP з'єднання встановлюється з конкретним пристроєм засобами TCP/IP. Тому адреса в пакеті Modbus найчастіше ігнорується, а широкомовне розсилання повідомлень не використовується. Однак адреса може бути потрібна, якщо з'єднання

встановлюється зі шлюзом, який, у свою чергу, виводить на мережу RS-485, щоб далі спілкуватися з пристроями вже мовою Modbus.

Контроль цілісності пакетів також забезпечується засобами протоколу TCP/IP, тому немає необхідності його Modbus-реалізації.

Поряд з адресою в заголовку пакета Modbus TCP є ряд додаткових полів:

– ID транзакції (або ID обміну). Це поле найчастіше заповнюється нулями. Поле необхідно для випадків, коли клієнтський пристрій надсилає кілька повідомлень, не чекаючи відповіді на попередні, щоб потім зв'язати відповіді із запитами;

– ID протоколу. Поле завжди заповнюється нулями, зарезервовано для майбутнього використання;

– довжина залишку пакета. Довжина частини пакета, що залишилася: адреси і PDU (коду функції і даних).

Рисунок 4.8 – Структура пакета протоколу Modbus TCP

4.3 Регістри та функції протоколу Modbus

4.3.1 Стандартна адресація регістрів Modbus

Оскільки Modbus призначений для роботи з промисловою автоматикою, обмін даними з Modbus-пристроями відбувається через регістри. Вони діляться на входи та виходи. Входи можна лише читати, а виходи – читати та писати. Бувають однобітні регістри Modbus для опису дискретних входів / виходів (Discrete Inputs та Coils) та 16-бітові регістри для аналогових входів / виходів (Input Registers та Holding Registers).

Доступ до регістрів здійснюється за допомогою 16-бітної адреси. Першому елементу в кожній групі регістрів відповідає адреса 0. Тобто адреса будь-якого регістру може приймати значення діапазону 0-65535 (0x0000-0xFFFF в HEX-

форматі). При цьому специфікація протоколу не визначає, що фізично представляють собою адресні простори і за якими внутрішніми адресами пристрою повинні бути доступні регістри. У випадку значення регістрів з однаковою адресою, але різними типами відрізняються один від одного.

В таблиці 4.1 подано стандартну адресацію регістрів Modbus.

Зазвичай, дискретні виходи починаються з адреси (осередка) 00001, а дискретні входи – з адреси 10001. Кожному з них потрібно 1 біт пам'яті. Вміст вхідних регістрів (в термінології контролерів вони іменувалися Contacts) можна тільки читати, в той час як вміст вихідних регістрів (в термінології ПЛК вони іменувалися Coils) можна і читати, і записувати.

Таблиця 4.1 – Таблиця розподілення регістрів Modbus

Адреса регістрів	Опис
00001 - 10000	1-бітні дискретні виходи (читання / запис) «Coils»
10001 - 20000	1-бітні дискретні входи (читання) «Discrete Inputs»
30001 - 40000	16-бітові аналогові входи (читання) «Input Registers»
40001 - 50000	16-бітові регістри зберігання (читання / запис) «Holding Registers»

Регістри аналогових входів і виходів є 16-розрядними. Їх адреси починаються з 30001 – це адреса першого аналогового входу (тільки читання, наприклад, для введення сигналів від датчика температури).

З адреси 40001 починається діапазон універсальних регістрів (читання і запис), які можуть слугувати також і аналоговими виходами.

Залежно від фірми-виробника ПЛК, ці регістри можуть бути внутрішніми регістрами, аналоговими входами, аналоговими виходами і навіть дискретними входами і виходами. Однак не всі функції працюють з адресами цих регістрів.

Перелік основних функцій протоколу Modbus подано у таблиці 4.2.

Незважаючи на те що кодам функцій відведений діапазон від 1 до 127, в якості призначених для загального користування кодів визначені приблизно 20 кодів. В цей же діапазон входять коди, призначення яких визначається кожним окремим користувачем. Слід мати на увазі, що багато Modbus-пристроїв підтримують тільки невеликі підмножини наявних кодів.

Таблиця 4.2 – Стандартні функції протоколу Modbus

Код	Роз-ряд-ність	Опис	Діапазон адрес входів / виходів
1	1	Read coils Читання поточного стану (ON / OFF) дискретних виходів	00001-10000
2	1	Read contacts Читання поточного стану (ON / OFF) дискретних входів	10001-20000
5	1	Write a single coil Зміна стану дискретного виходу в ON або OFF	00001-10000
15	1	Write multiple coils Зміна стану (ON / OFF) декількох дискретних виходів	00001-10000
3	16	Read holding registers Читання регістрів зберігання	40001-50000
4	16	Read input registers Читання вхідних регістрів	30001-40000
6	16	Write single register Запис одного регістра	40001-50000
16	16	Write multiple registers Запис декількох регістрів	40001-50000
22	16	Mask write register Маскований запис регістра	40001-50000
23	16	Read/write multiple registers Читання / запис декількох регістрів	40001-50000
24	16	Read FIFO queue Читання вмісту черги FIFO	40001-50000

4.3.2 Опис функції читання вихідних контактів (дискретних виходів) (01)

Функція (01) дозволяє користувачеві отримати статус вихідних контактів. Широкомовний режим не підтримується.

Крім полів адреси та функції, повідомлення вимагає, щоб інформаційне поле містило логічну адресу початкового біту і кількість бітів, статус яких необхідно отримати.

Адресація дозволяє отримати за один запит до 2000 логічних осередків. Однак, деякі прилади мають обмеження на максимальне число бітів, статус яких можна отримати за один запит. Біти нумеруються з нуля (біт 1 = 0, біт 2 = 1 і т.д.).

Припустимо, нам потрібно звернутися до підлеглого пристрою з адресою 11 та прочитати 19 його Coil-регістрів з номерами 20-38. Адресація дискретних виходів починається з 0, тому адреса першого потрібного нам виходу буде 0x13 (це 19 у HEX-системі). Необхідна для читання кількість виходів також дорівнюватиме 0x13 (для читання запитано 19 виходів). Адреса пристрою 0x11 (число 17 в HEX-форматі) та код функції 01. Контрольна сума формується за алгоритмом CRC-16 на основі інших полів пакета.

У таблиці 4.3 представлений запит на читання дискретних виходів 0020-0038 з приладу з адресою 17 (0x11).

Таблиця 4.3 – Приклад запиту для читання дискретних виходів 0020 – 0038

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
01	Код функції
00	Адреса початкового біту, Hi байт
13	Адреса початкового біту, Lo байт
00	Кількість дискретних виходів, Hi байт
13	Кількість дискретних виходів, Lo байт
8E	Контрольна сума CRC
92	Контрольна сума CRC

У таблиці 4.4 поданий приклад відповіді на запит читання дискретних виходів 0020-0038

Таблиця 4.4 – Приклад повідомлення у відповідь

Байт	Опис полів відповіді
11	Адреса підлеглого пристрою
01	Код функції
03	Кількість байт в полі даних
A3	Статус вихідних контактів, перший байт
24	Статус вихідних контактів, другий байт
07	Статус вихідних контактів, третій байт
94	Контрольна сума CRC
3E	Контрольна сума CRC

На рис. 4.9 подано приклад обміну повідомленнями з використанням функції 01 (читання дискретних виходів).

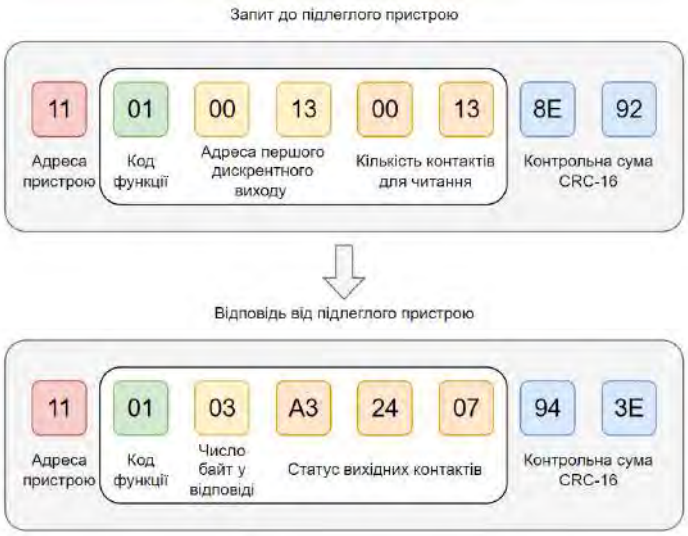


Рисунок 4.9 – Приклад обміну повідомленнями при використанні функції 01 (читання дискретних виходів)

Відповідь містить адресу, код функції, число байт в полі даних, дані і контрольну суму. Дані в полі даних у відповіді представлені у вигляді один біт на кожен осередок.

Молодший значущий біт першого байту поля даних містить перший осередок, за яким слідують інші. Якщо число осередків не ділиться на 8, то інші біти заповнюються нулями в порядку від старших бітів до молодших.

Статус осередків 20-27 дорівнює 0xA3 = 1010 0011. Читаючи зліва направо, бачимо, що осередки 27, 25, 21 і 20 встановлені. Інші дані розбираються так само. Так як було запрошено число регістрів, що не ділиться на 8, старші п'ять біт в останньому байті даних (0x07) заповнені нулями.

Принцип побудови послідовності відповідних байтів у зворотному повідомленні подано на рис. 4.10.

На даному прикладі подано випадок, коли робиться запит на читання десяти вхідних контактів, починаючи з третього входу (адреса входу DI2). Виділена область в модулі виведення дискретних сигналів підлягає зчитуванню.

На рис. 4.10 подано принцип збирання бітів у байти відповіді. Молодші біти переходять до правої частини результуючого байту, а старші – до лівої.

Якщо запит прийшов на кількість бітів, що менша за кількість бітів в байті, біти, що залишаються, заповнюються нулями (другий байт відповіді 0x02).

Запит обслуговується в кінці робочого циклу приладу, тому потрібно мати на увазі, що дані у відповідному повідомленні відображають стан регістрів на той момент.

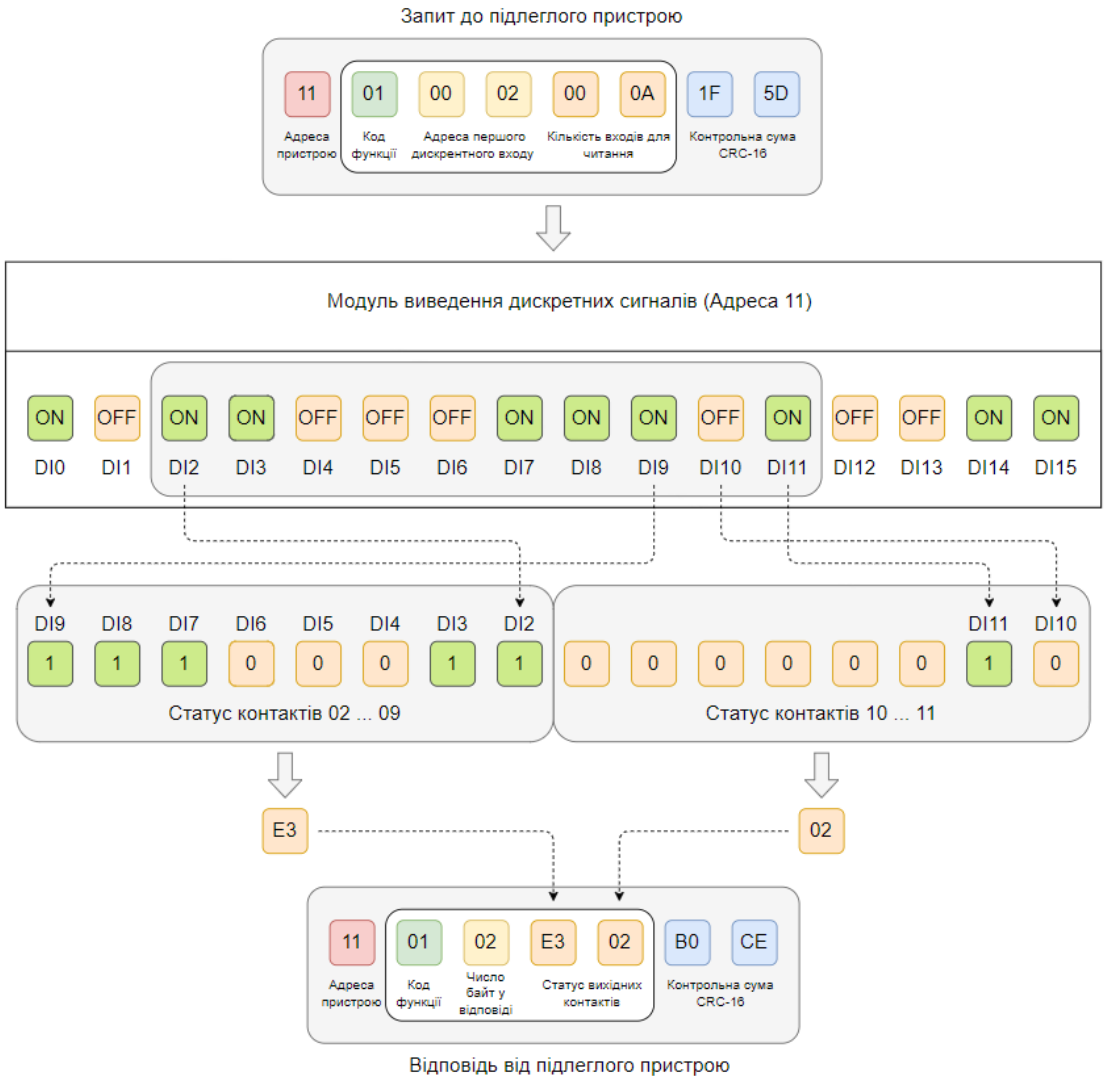


Рисунок 4.10 – Принцип побудови послідовності відповідних байтів у зворотному повідомленні

4.3.3 Функція читання дискретних входів (02)

Ця функція дозволяє користувачеві отримати стан (УВІМКН / ВИМКН) вхідних дискретних ліній пристрою, що адресується. Циклічний запит не підтримується. На додаток до адреси пристрою і номеру функції, запит вимагає, щоб інформаційне поле містило початкову адресу і кількість необхідних ліній.

Адресація дозволяє отримати за один запит до 2000 ліній. Однак, деякі пристрої мають обмеження на максимальну кількість ліній, одержуваних за один запит. Вхідні лінії нумеруються з нуля (10001 = 0, 10002 = 1 і т.д.).

У таблиці 4.5 подано приклад запиту на читання дискретних входів 10197-10218 з пристрою з адресою 17 (0x11).

Таблиця 4.5 – Приклад запиту на читання дискретних входів 10197-10218

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
02	Код функції
00	Адреса початкового біту, Hi байт
C4	Адреса початкового біту, Lo байт
00	Кількість дискретних входів, Hi байт
16	Кількість дискретних входів, Lo байт
BA	Контрольна сума CRC
A9	Контрольна сума CRC

Приклад відповіді на цей запит подано в таблиці 4.6.

Відповідь містить адресу пристрою, код функції, кількість байт даних, дані і поле контрольної суми.

Дані упаковані по біту на кожен вхід (1 = ON, 0 = OFF). Молодший біт першого байту містить значення першого входу, що адресується, за яким слідують інші. Якщо кількість запитаних входів не кратне 8, то інші біти заповнюються нулями. Кількість байт даних завжди визначається як кількість RTU даних.

Так як пристрої обслуговує запит в кінці робочого циклу, дані у відповіді відображають стан входів на даний момент. Деякі пристрої мають обмеження на максимальну кількість входів, запитуваних за один запит.

Статус входів 10197-10204 = 0xAC = 1010 1100. Читаючи зліва направо, бачимо, що входи 10204, 10202, 10200 і 10199 в стані ON. Всі інші байти даних розпаковуються аналогічно.

Так як була запрошена інформація про стан 22 вхідних контактів, останній байт даних (35h = 0011 0101) містить інформацію тільки про 6 входів (10213-10218) замість восьми. Два останніх біта заповнюються нулями.

Таблиця 4.6 – Приклад повідомлення у відповідь на функцію 02

Байт	Опис полів відповіді
11	Адреса підлеглого пристрою
02	Код функції
03	Кількість байт в полі даних
AC	Статус вхідних контактів, перший байт
DB	Статус вхідних контактів, другий байт
35	Статус вхідних контактів, третій байт
20	Контрольна сума CRC
18	Контрольна сума CRC

Приклад обміну повідомленнями під час використання функції 02 (читання дискретних входів) подано на рис. 4.11 та 4.12.

Рисунок 4.11 – Приклад запиту для читання дискретних входів 10197-10204

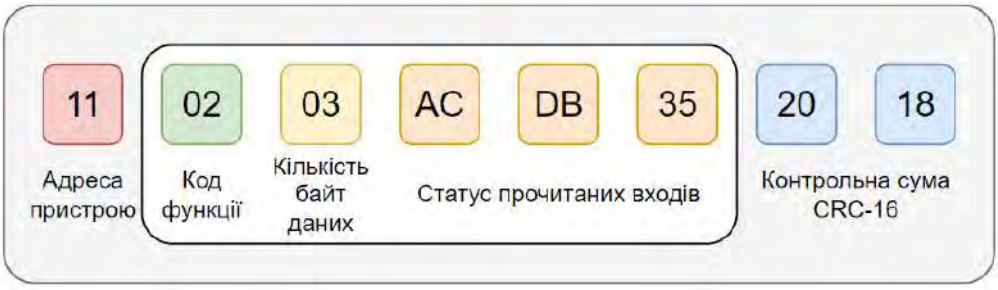


Рисунок 4.12 – Приклад відповіді на запит читання дискретних входів

4.3.4 Функція читання регістрів зберігання (03)

Ця функція дозволяє отримати бінарний вміст 16-ти розрядних регістрів підлеглого пристрою. Адресація дозволяє отримати за кожен запит до 125 регістрів. Однак, деякі пристрої мають обмеження на максимальну кількість

регістрів, одержуваних за один запит. Регістри нумеруються з нуля (40001 = 0, 40002 = 1 і т.д.). Широкомовний режим не допускається.

У таблиці 4.7 подано приклад запиту на читання регістрів 40108-40110 з пристрою з адресою 17 (0x11).

Таблиця 4.7 – Приклад повідомлення у відповідь на функцію 03

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
03	Код функції
00	Адреса першого байту, Hi байт
6B	Адреса першого байту, Lo байт
00	Число регістрів для читання, Hi байт
03	Число регістрів для читання, Lo байт
76	Контрольна сума CRC
87	Контрольна сума CRC

Пристрій, що адресується посилає у відповіді свою адресу, код виконаної функції і інформаційне поле.

Інформаційне поле містить 2 байти, що описують кількість байт даних, що повертаються. Довжина кожного регістру даних – 2 байти. Перший байт даних в посилці є старшим байтом регістру, другий – молодшим.

Так як пристрій зазвичай обслуговує запит в кінці свого робочого циклу, дані у відповіді відображають вміст регістрів в даний момент.

Деякі пристрої обмежують кількість регістрів, переданих за один запит. У цьому випадку для отримання, більшого числа регістрів, необхідно виконати кілька послідовних запитів.

У таблиці 4.8 представлений приклад відповіді на запит читання регістрів 40108-40110, що мають вміст, відповідно, 555, 0, 100, з пристрою з адресою 17 (0x11).

На рис. 4.13 подано приклад контролера «МС-1», що управляє частотою обертів вентилятора. На даному прикладі розглянемо принцип моніторингу стану системи управління вентиляцією за допомогою протоколу Modbus.

Таблиця 4.8 – Приклад відповіді на функцію 03

Байт	Опис полів відповіді
11	Адреса підлеглого пристрою
03	Код функції
06	Кількість байт в полі даних
02	Значення першого байту даних, Hi байт
2B	Значення першого байту даних, Lo байт
00	Значення другого байту даних, Hi байт
00	Значення другого байту даних, Lo байт
00	Значення третього байту даних, Hi байт
64	Значення третього байту даних, Lo байт
C8	Контрольна сума CRC
BA	Контрольна сума CRC

В пристрої, що зображений на рис. 4.13 присутні п’ять внутрішніх регістрів для зберігання поточних даних роботи пристрою управляння вентилятором.

Регістр 40001 зберігає поточну швидкість двигуна вентилятора. Отримавши значення з цього регістру можна дізнатися в якому стані зараз знаходиться двигун та як швидко він обертається. Інформація в дому регістрі оновлюється на основі даних від датчика обертів (енкодеру).

В регістрі 40002 зберігається поточний час роботи двигуна вентилятора з моменту останнього увімкнення. Після зупинки, та нового старту, це значення скидається та оновлюється. Значення даного регістру збільшується на одиницю кожну секунду. Максимальне значення, що може зберігатися в регістрі – 65535 с.

Регістр 40003 зберігає номінальну, або задану швидкість роботи двигуна вентилятора. Це значення автоматично змінюється при зміні режиму роботи вентилятора за допомогою кнопки, або заноситься в даний регістр за допомогою протоколу Modbus через промислову мережу. Значення в регістрі задається в форматі «число обертів за хвилину», наприклад, 500 об/хв.

Регістр 40004 зберігає поточний режим роботи вентилятора (число в діапазоні від 0 до 4). Всього передбачено 5 режимів, або 5 швидкостей:

– 0: двигун вимкнено, швидкість 0 об/хв.;

– 1: двигун обертається з мінімальною швидкістю 150 об/хв.;

– 2: двигун обертається зі швидкістю 500 об/хв.;

– 3: двигун обертається зі швидкістю 1000 об/хв.;

– 4: двигун обертається з максимальною швидкістю 1500 об/хв.

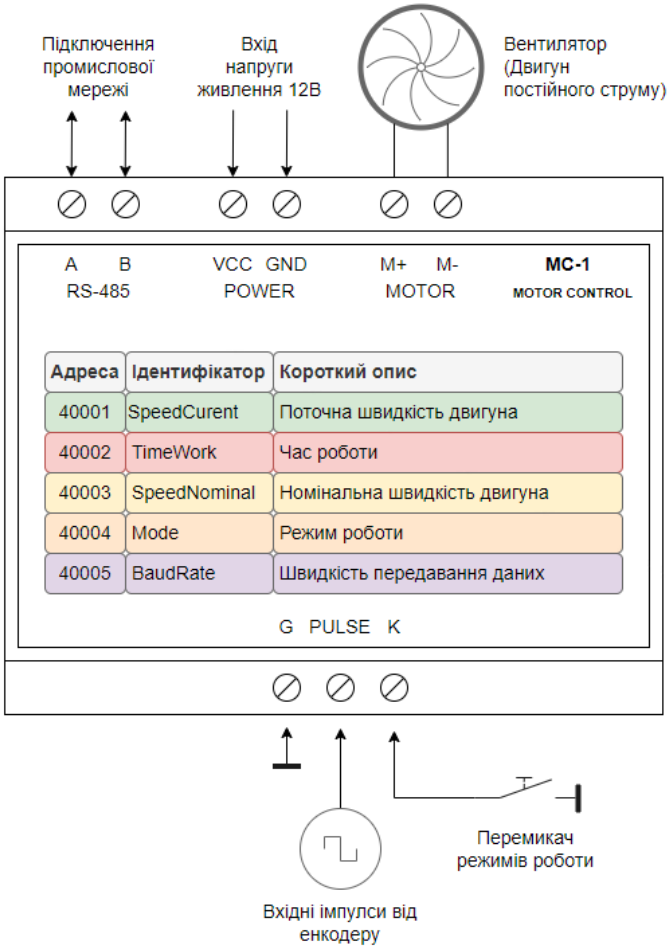


Рисунок 4.13 – Приклад контролера «МС-1», що управляє частотою обертів вентилятора

Регістр 40005 зберігає налаштування швидкості передачі даних в мережі Modbus. Можливі значення: 1200, 2400, 4800, 9600, 19200 та 38400 біт/с.

Розглянемо приклад перевірки значення поточної швидкості роботи двигуна. Виходячи з рис. 4.13, поточна швидкість зберігається в регістрі 40001. Для читання цього значення формуємо запит, та отримуємо відповідь (рис. 4.14).

В запиті в якості початкової адреси вказуємо число «00 00» тому, що 40001 – це перший, тобто нульовий регістр (відлік починається з нуля). Після початкової адреси вказуємо кількість регістрів, інформацію з яких мі хочемо отримати. В даному прикладі – це один регістр. Таким чином, наступні два байти будуть мати значення «00 01» (відповідно до таблиці 4.7).

В результаті вдалого обміну даними головний пристрій отримає наступне повідомлення:

11 03 02 05 E1 BA 9F

В цьому повідомленні, після номеру функції «03» йде кількість переданих байт («02») та самі байти даних («05 E1»).

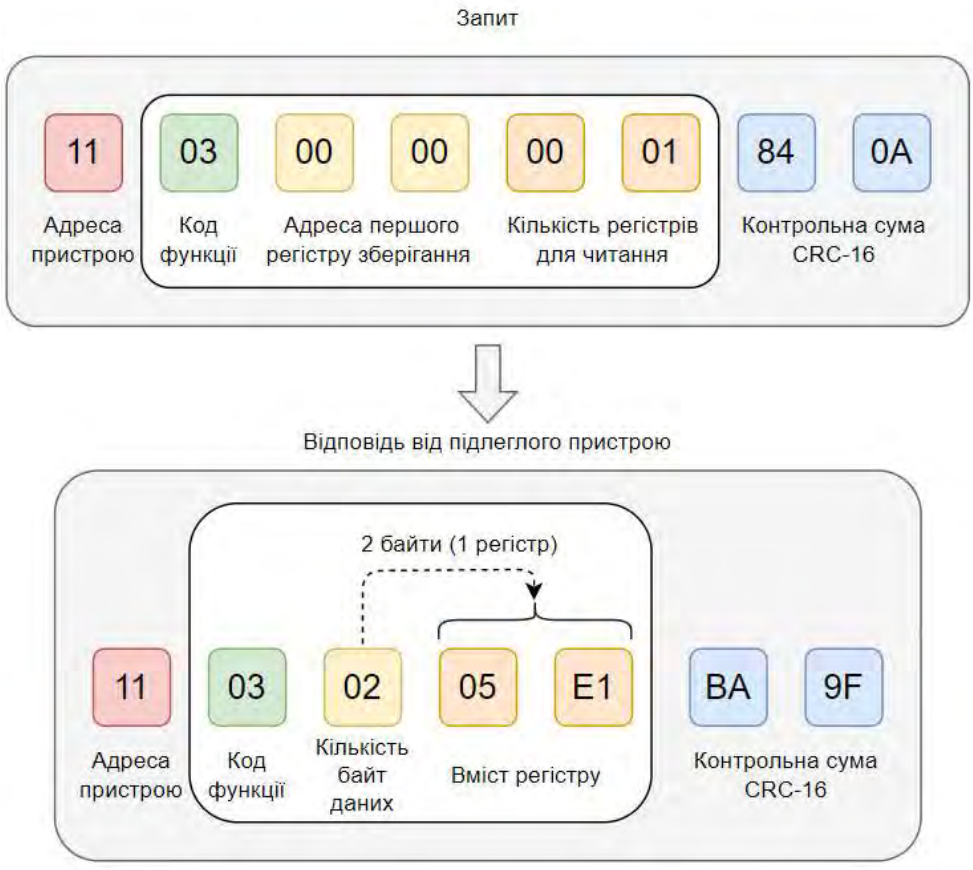


Рисунок 4.14 – Приклад обміну даним між пристроями в процесі читання значення регістру 40001

Для того , щоб дізнатися реальну швидкість обертів двигуна вентилятора необхідно перевести число 05E1 із шістнадцятирічного формату запису в десятковий (рис. 4.15).

Таким чином, ми бачимо, що поточне значення швидкості оберту становить 1505 об/хв.

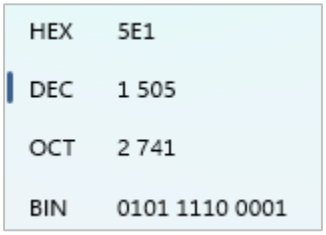


Рисунок 4.15 – Представлення числа 05E1 в десятковому вигляді

Для того, щоб дізнатись, чи відповідає така швидкість поточному режиму вентилятора, та визначити задану номінальну швидкість, необхідно прочитати вміст регістрів зберігання за адресами 40003 та 40004.

На рис. 4.16 показано сформований запит до пристрою МС-1.

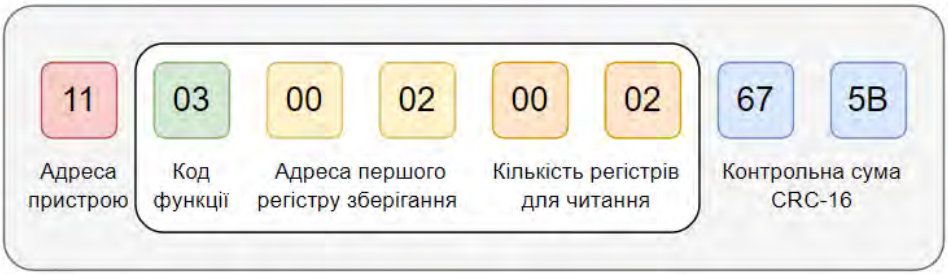


Рисунок 4.16 – Запит до контролера «МС-1»

В запиті вказується адреса першого регістра для читання – 0002, що відповідає адресі 40003. Кількість регістрів, що необхідно прочитати – 2, тому наступні два байти в запиті: 0002. В результаті читання вказаних регістрів отримуємо таку відповідь (рис. 4.17).

Рисунок 4.17 – Відповідь від контролера МС-1

Проведемо аналіз отриманої відповіді у відповідності до таблиці 4.8. Відповідь містить номер функції (03) та інформацію про кількість байтів даних, що містять інформацію з регістрів зберігання. Таких байтів – чотири (по два байти на один 16-ти розрядний регістр).

Виходячи з отриманої інформації, перший з прочитаних регістрів (адреса в пристрої МС-1 40003) містить число 05DC. Відповідно таблиці призначення регістрів (рис. 4.12), ми отримали номінальну швидкість двигуна вентилятора. Якщо перевести це число в десятковий формат, отримаємо число 1500 (рис. 4.18).

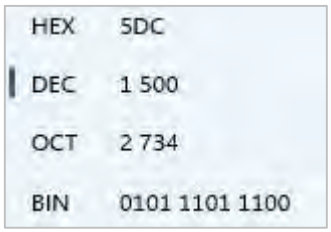


Рисунок 4.18 – Представлення числа 05DC в десятковому вигляді

Таким чином, ми бачимо, що номінальне значення швидкості оберту двигуна вентилятора становить 1500 об/хв. Порівнюючи з поточною швидкістю двигуна, що була отримана в попередньому сеансі обміну даними з пристроєм керування (1505 об/хв), можна бачити, що різниця невелика, та становить 0,3 %. Таке відхилення в швидкості роботи двигуна є припустимим та лежить в межах норми.

Другий байт містить число, що відповідає режиму роботи вентилятора. Виходячи з опису роботи контролеру МС-1, швидкості 1500 об/хв повинен відповідати 4 режим роботи. Якщо подивитись на рис. 4.17, то саме це число й міститься у відповіді (0004).

Таким чином, сеанс обміну інформацією з контролером управління вентилятором МС-1 показав, що прилад працює в штатному режимі, відхилень в роботі не виявлено.

4.3.5 Запис одного регістру зберігання (06)

Ця функція використовується для запису одного регістру зберігання у віддаленому пристрої. PDU запиту вказує адресу регістру, який потрібно записати. Адреси регістрів починаються з нуля, тому регістр під номером 1 адресується як 0.

В таблиці 4.9 подано приклад запиту для запису в регістр 40136 числа 926 (0x039E). Адреса пристрою 17 (0x11).

Таблиця 4.9 – Приклад запису в регістр 40136 числа 926

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
06	Код функції
00	Адреса першого байту, Hi байт
87	Адреса другого байту, Lo байт
03	Значення байту даних, Hi байт
9E	Значення байту даних, Lo байт
5A	Контрольна сума CRC
D3	Контрольна сума CRC

У випадку вдалого виконання запиту у відповідь буде надіслане повідомлення, що ідентичне запиту. Приклад відповіді подано в таблиці 4.10.

Таблиця 4.10 – Приклад відповіді на запит запису в регістр 40136 числа 926

Байт	Опис полів відповіді
11	Адреса підлеглого пристрою
06	Код функції
00	Адреса першого байту, Hi байт
87	Адреса другого байту, Lo байт
03	Значення байту даних, Hi байт
9E	Значення байту даних, Lo байт
5A	Контрольна сума CRC
D3	Контрольна сума CRC

Розглянемо приклад застосування даної функції протоколу Modbus на практиці. Наприклад, необхідно змінити режим роботи контролера МС-1, що керує роботою вентилятора.

Для зміни режиму роботи необхідно змінити зміст регістру 40004 (рис. 4.13). Якщо ми хочемо перевести вентилятор в економічний режим, нам

потрібно змінити режим роботи на «1». В даному режимі двигун буде обертатися з мінімальною швидкістю 150 об/хв.

На рис. 4.19 подано приклад запиту для переведення вентилятора в режим роботи №1.

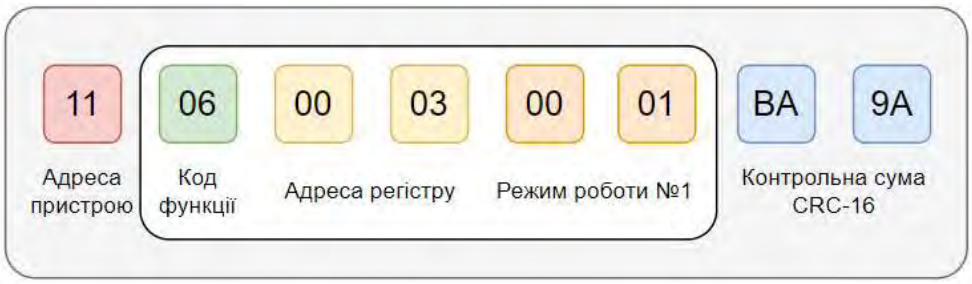


Рисунок 4.19 – Приклад запиту для зміни режиму роботи двигуна вентилятора

Очікувана відповідь повторює запит (рис. 4.20).

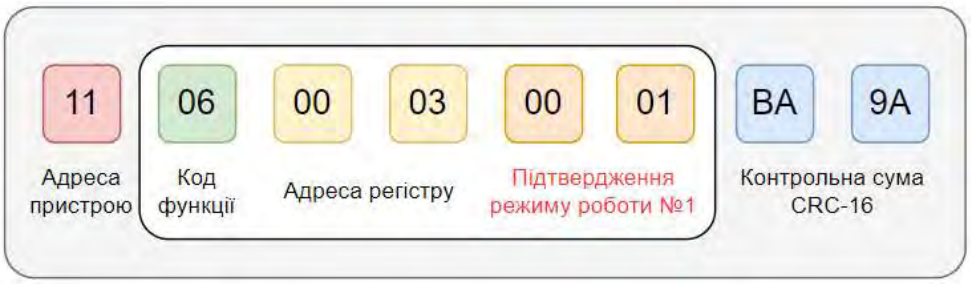


Рисунок 4.20 – Підтвердження запису в регістр 40003 режиму роботи двигуна вентилятора №1

4.3.6 Запис декількох регістрів зберігання (0x10)

Часто виникають задачі модифікації одразу декількох регістрів зберігання. Для цього в протоколі Modbus передбачена функція з номером 16 (0x10).

В таблиці 4.11 подано приклад запиту для запису в регістри 40136 та 40137 значень 0x00a0 та 0x0102. Адреса пристрою 17 (0x11).

Повідомлення дозволяє записувати регістри з максимальною логічною адресою до 0xFFFF. Не використовувані старші біти адреси регістрів повинні заповнюватися нулями. Якщо використовується широкомовний запит, то вміст поля даних записується у всі пристрої, підключені до шини.

Таблиця 4.11 – Приклад запиту для запису значень в регістри 40136 та 40137

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
10	Код функції
00	Адреса першого байту, Hi байт
87	Адреса першого байту, Lo байт
00	Кількість регістрів для запису, Hi байт
02	Кількість регістрів для запису, Lo байт
04	Кількість байт в полі даних
00	Значення першого байту даних, Lo байт
0A	Значення першого байту даних, Lo байт
01	Значення другого байту даних, Hi байт
02	Значення другого байту даних, Lo байт
4E	Контрольна сума CRC
BA	Контрольна сума CRC

У випадку вдалого виконання запиту у відповідь буде надіслане повідомлення, що містить адресу першого байту, та кількість байт даних, що було записано. Приклад відповіді подано в таблиці 4.12.

Таблиця 4.12 – Відповідь на запит щодо запису даних в два регістри зберігання

Байт	Опис полів відповіді
11	Адреса підлеглого пристрою
10	Код функції
00	Адреса першого байту, Hi байт
87	Адреса другого байту, Lo байт
00	Кількість записаних регістрів, Hi байт
02	Кількість записаних регістрів, Lo байт
F3	Контрольна сума CRC
71	Контрольна сума CRC

Розглянемо приклад застосування даної функції протоколу Modbus на практиці. Наприклад, необхідно за один сеанс обміну інформацією змінити режим роботи контролеру МС-1, що керує роботою вентилятора, та швидкість передачі даних в мережі.

Виходячи з рис. 4.13, режим роботи пристрою змінюється за допомогою регістра 40004, до якого записується число в діапазоні від 0 до 4. Швидкість передачі даних задається в залежності від значення, що міститься в регістрі 40005.

На рис. 4.21 подано приклад запиту, в якому міститься інформація про новий режим роботи (режим №2), та про нову швидкість передачі даних (19200 біт/с).

Рисунок 4.21 – Приклад запиту для зміни режиму роботи та швидкості передачі даних

На рис. 4.22 подано отриману відповідь від контролера.

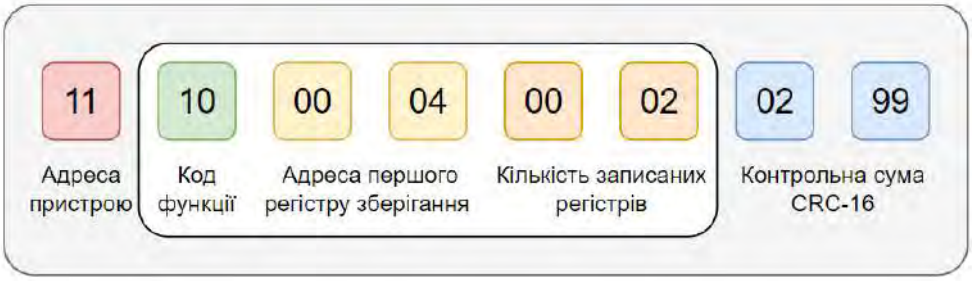


Рисунок 4.22 – Отримана відповідь від контролера

З рис. 4.22 можна бачити, що в контролері МС-1 було змінено два регістри, починаючи з адреси 40005.

4.3.7 Зміна стану одного вихідного контакту (05)

Це повідомлення модифікує один логічний осередок, що уявляє собою один дискретний вихід контролера. Осередки нумеруються з нуля (осередок 1 = 0, осередок 2 = 1 і т.д.).

Для увімкнення дискретного виходу необхідно передати число 0xFF00. Ця команда встановлює певний осередок в 1. Якщо необхідно вимкнути дискретний вихід, необхідно передати число 0x0000. Ця команда переведе певний осередок в стан логічного 0. Інші передані числа не впливають на вміст осередка, що адресується. Ця функція може використовуватися в широкомовному режимі.

У таблиці 4.13 подано приклад встановлення в 1 дискретного виходу з адресою 0173 (0x00AD) в пристрій з адресою 17 (0x11).

Таблиця 4.13 – Приклад встановлення в стан логічної 1 осередка 0173

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
05	Код функції
00	Адреса дискретного виходу, Hi байт
AD	Адреса дискретного виходу, Lo байт
FF	Ознака встановлення (0xFF), або скидання (0x00)
00	Завжди містить значення 0x00
1F	Контрольна сума CRC
4B	Контрольна сума CRC

У відповідь повинне надійти повідомлення, яке повністю співпадає з запитом.

4.3.8 Зміна стану декількох вихідних контактів (0F)

Цей код функції використовується для примусового ввімкнення або вимкнення кожного вихідного контакту (котушки) в межах адресного простору у віддаленому пристрої. PDU запиту вказує посилання на певні контакти, які потрібно примусово встановити, або вимкнути. Контакти адресуються починаючи з нуля. Тому котушка під номером 1 адресується як 0.

Необхідні стани On / Off визначаються вмістом поля даних запиту. Логічна «1» у бітовій позиції поля вимагає, щоб відповідний вихід був увімкнений. Логічний «0» вимагає, щоб його було вимкнено.

У таблиці 4.14 наведено приклад зміни стану 9 дискретних виходів з адресами починаючи з 00028 до 00036 для віддаленого пристрою з адресою 11 (0x0B).

Таблиця 4.14 – Приклад зміни стану 9 дискретних виходів

Байт	Опис полів запиту
0B	Адреса підлеглого пристрою
0F	Код функції
00	Початкова адреса дискретного виходу, Hi байт
1B	Початкова адреса дискретного виходу, Lo байт
00	Кількість виходів, стани яких потрібно змінити, Hi байт
09	Кількість виходів, стани яких потрібно змінити, Lo байт
02	Кількість байт даних
4D	Перший байт даних
01	Другий байт даних
6C	Контрольна сума CRC
A7	Контрольна сума CRC

Принцип побудови байт даних подано на рис. 4.23.

Кількість байт даних визначається таким чином:

9 дискретних виходів = 1 байт + 1 біт + 7 порожніх біт = 2 байти.

Далі збираються байти даних. Дискретні виходи 35-28 мають наступну комбінацію: 0100 1101. Дискретний вихід 36 повинен бути увімкнений (логічна 1). Після додавання 7-ми порожніх біт, отримаємо комбінацію: 0000 0001. Таким чином, два байти даних будуть мати наступне значення: 4D 01.

Як можна бачити з рис. 4.23, старші біти містять старші змінні вихідних контактів. З поданого прикладу видно, що котушка 28 увімкнена (логічна 1), а котушка 35 вимкнена (логічний 0). Останнє поле даних містить статус лише

одного дискретного виходу (логічна 1). Невикористані біти в останньому байті даних заповнюються нулями – це тримачі простору.

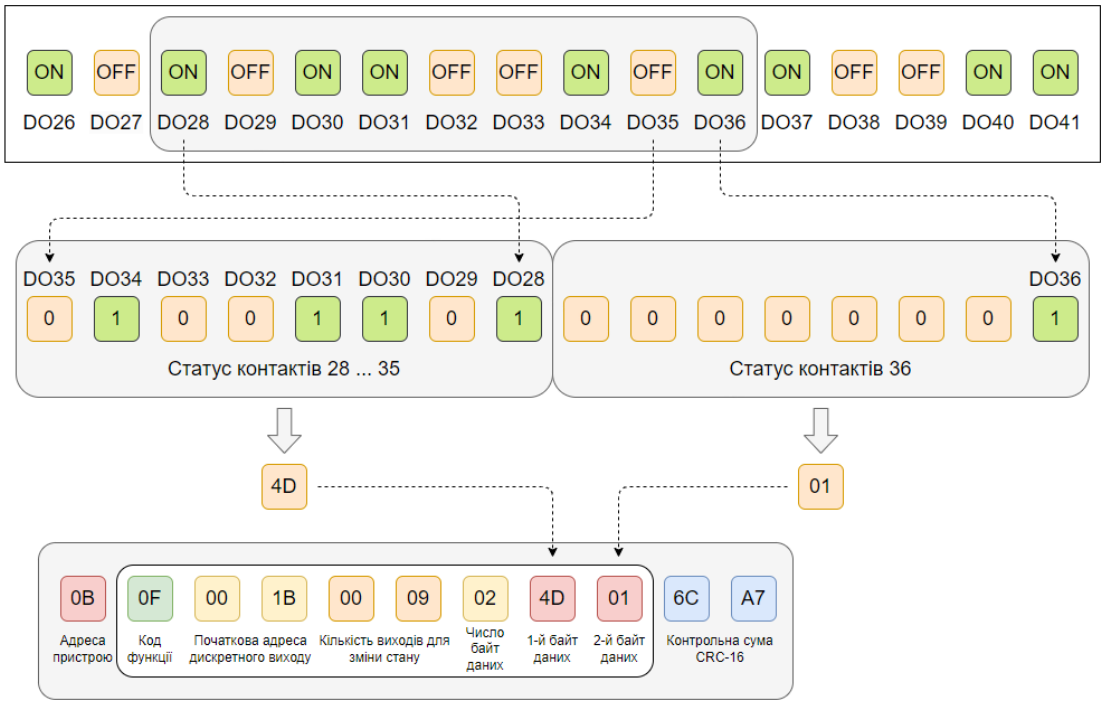


Рисунок 4.23 – Принцип побудови байт даних для формування повідомлення зміни декількох дискретних виходів

Відповідь від підлеглого пристрою містить адресу розміщення першого дискретного виходу та підтвердження кількості записаних дискретних виходів (табл. 4.15).

Таблиця 4.15 – Відповідь на запит щодо зміни стану декількох вихідних контактів

Байт	Опис полів відповіді
0B	Адреса підлеглого пристрою
0F	Код функції
00	Адреса першого байту, Hi байт
1B	Адреса другого байту, Lo байт
00	Кількість записаних дискретних виходів, Hi байт
1B	Кількість записаних дискретних виходів, Lo байт
E5	Контрольна сума CRC
60	Контрольна сума CRC

4.3.9 Читання стану вхідних регістрів (04)

Вхідні регістри в промислових контролерах зазвичай використовуються для запису даних, що отримуються від аналого-цифрових перетворювачів. Це можуть бути значення виміряного струму, напруги або, наприклад, температури. Вхідні регістри 16-розрядні. Якщо значення параметру, що вимірюється, перевищують 16 біт (наприклад, АЦП розрядністю 24 біти), то в пам'яті ПЛК вони займатимуть два регістри.

В протоколі Modbus для читання вхідних регістрів використовується функція 0x04. Вона використовується для безперервного читання від 1 до 125 вхідних регістрів з віддаленого пристрою. PDU запиту вказує початкову адресу регістра та кількість регістрів. В PDU адресація регістрів починаються з нуля. Тому вхідний регістр під номером 1 адресується як 0.

У таблиці 4.16 представлений приклад запиту на читання значення АЦП з вхідного регістру 30011 з пристрою з адресою 17 (0x11).

Таблиця 4.16 – Приклад запиту на читання значення АЦП

Байт	Опис полів запиту
11	Адреса підлеглого пристрою
04	Код функції
00	Адреса першого байту вхідного регістру, Hi байт
0A	Адреса першого байту вхідного регістру, Lo байт
00	Число регістрів для читання, Hi байт
01	Число регістрів для читання, Lo байт
13	Контрольна сума CRC
58	Контрольна сума CRC

Пристрій, що адресується посилає у відповіді свою адресу, код виконаної функції і інформаційне поле.

Для даного прикладу інформаційне поле містить 2 байти, що описують кількість байт даних, що повертаються. Довжина кожного регістру даних – 2 байти. Перший байт даних в посиланні є старшим байтом регістру, другий – молодшим.

У таблиці 4.17 подано приклад відповіді на запит читання вхідного регістру 30011, в якому міститься число 0x102F (4143), що у двійковому форматі має наступний вигляд – 0001 0000 0010 1111.

Таблиця 4.17 – Приклад відповіді на функцію 04

Байт	Опис полів відповіді
11	Адреса підлеглого пристрою
04	Код функції
02	Кількість байт в полі даних
10	Значення першого байту даних, Hi байт
2F	Значення першого байту даних, Lo байт
34	Контрольна сума CRC
EF	Контрольна сума CRC

4.4 Контрольні запитання

1. Що таке протокол Modbus і яке його основне призначення в промислових мережах?
2. Які існують різновиди протоколу Modbus і в чому їх основні відмінності?
3. Які особливості має Modbus RTU (Remote Terminal Unit)?
4. У чому полягає специфіка передачі даних у форматі Modbus ASCII?
5. Що таке Modbus TCP і як він використовується в Ethernet-мережах?
6. Як організовується обмін даними за протоколом Modbus між головним і підлеглими пристроями?
7. Яка структура повідомлення у форматі Modbus RTU?
8. Які типи регістрів визначено в Modbus і які з них призначені для зберігання, читання або запису даних?
9. Яке значення має стандартна адресація регістрів у протоколі Modbus?
10. Що таке функція 01 і як вона працює?
11. Як реалізується функція 02?
12. Що таке функція 03 і в яких випадках вона застосовується?
13. Як працює функція 06?

14. У чому полягає відмінність між функціями запису одного (06) та декількох регістрів зберігання (0x10)?

15. Як реалізується функція 05?

16. Які основні режими роботи протоколу Modbus та які їх відмінності?

17. Який принцип закладено у роботу протоколу Modbus?

18. Які функції використовуються для запису даних у пристрій?

19. Як можна дізнатися про стан вихідних контактів? Наведіть приклад запиту.

20. Як можна змінити стан вихідних контактів? Наведіть приклад запиту.

21. Як можна отримати дані про стан датчика температури, який підключено до модулю вводу аналогових сигналів?

22. Як можна дізнатися про стан дискретних входів? Наведіть приклад запиту.

23. Як змінити стан декількох дискретних виходів за один запит? Наведіть приклад запиту.

24. Які типи повідомлень використовуються в протоколі Modbus, і які їх особливості?

25. Які основні алгоритми помилкового контролю в протоколі Modbus, та як вони використовуються для виявлення та корекції помилок передачі даних?

26. Які основні принципи роботи пристроїв, які використовують протокол Modbus, та як вони можуть бути інтегровані в системи контролю та управління виробництвом?

5 ВИКОРИСТАННЯ ПРОТОКОЛУ MODBUS ДЛЯ КЕРУВАННЯ ЗАСОБАМИ АВТОМАТИЗАЦІЇ

5.1 Програмне забезпечення для тестування та налагодження пристроїв і мереж на базі Modbus

5.1.1 Тестування пристроїв із підтримкою Modbus RTU в рамках процесу розробки

Як в процесі розробки, так і в процесі налагодження пристроїв з підтримкою протоколу Modbus RTU необхідно мати спеціалізоване програмне забезпечення та технічні засоби. З технічних засобів найпростіший варіант – це перетворювач RS-485/USB. Прикладами таких пристроїв можуть бути:

– захищений модуль адаптера USB-RS485 [16];

– перетворювач RS-485 на USB [17].

Зовнішній вигляд захищеного модуля адаптера USB-RS485 подано на рис. 5.1.

Рисунок 5.1 – Зовнішній вигляд захищеного модуля адаптера USB-RS485

Модуль адаптера USB-RS485 з входом USB-B призначений для підключення мережі з інтерфейсом RS-485 до персонального або міні-комп'ютера. У модуль вбудований захист виходу від перенапруги і статичної електрики.

Характеристики пристрою:

– інтерфейс підключення до комп'ютера: USB 2.0 B Female;

– вихідний інтерфейс: RS485;

– контроль напрямку – автоматичне управління потоком даних, автоматична ідентифікація і управління напрямком передачі даних;

– швидкість передачі даних: 300-9216000 біт/с, автоматичне визначення швидкості послідовних даних;

– можливості підключення пристроїв: підтримує багатоточкове підключення, кожен конвертер дозволяє підключати до 32 пристроїв RS-485;

– сумісність: Windows 98/ME/2000 /XP/WIN7/Vista, Linux, Mac;

– підтримка пристроїв з інтерфейсом RS-485: камери спостереження, пристрої захоплення відео, пристрої для зчитування відбитків пальців, PBX, CNC верстати, SCM тощо.

Перетворювач RS-485 в USB не має засобів гальванічного захисту та може застосовуватись в лабораторному практикумі для дослідження властивостей протоколу Modbus. Зовнішній вигляд пристрою подано на рис. 5.2.

Рисунок 5.2 – Зовнішній вигляд перетворювача RS-485 в USB

В процесі розробки пристроїв з підтримкою Modbus RTU, найчастіше потрібно реалізувати функцію Slave, оскільки в основному це різні датчики, керовані реле, модулі вводу/виводу тощо, Master-пристрої створюються рідше. У мережах автоматизації в якості майстра, зазвичай виступає контролер, а він, як правило, вже має реалізацію Modbus-стеку, або OPC Server/SCADA система, укомплектована Modbus-драйвером.

Другим важливим моментом під час розробки Modbus-пристроїв є тестування.

На початкових етапах корисним інструментом є Modbus-термінал. За допомогою нього можна вручну сконструювати запит, надіслати його та проаналізувати відповідь. Існують термінали в чистому вигляді, наприклад

SmartTerminal, Access Port, термінали з підтримкою Modbus RTU – Termite від S2-Team.

В процесі розробки нерідко виникає ситуація, коли пристрій приймає запит і відповідає на нього (це можна зрозуміти або за світлодіодами приймання / передавання пакетів, якщо ви їх передбачили в конструкції, або через відладчик, поставивши breakpoint у потрібному місці), а в терміналі або в іншій спеціалізованій програмі, дані не відображаються. У такому випадку для розуміння проблеми знадобиться сніффер для послідовного порту, за допомогою протоколу Modbus. Як приклад можна навести Free Serial Analyzer, COM Port Toolkit.

Надалі, потрібно не тільки перевіряти, чи працює пристрій в принципі (тобто коректно відповідає на запити), а й визначати напрацювання на відмову за допомогою тривалого тестування. Важливими аспектами тут є підтримка авто-опитування зі змінним навантаженням (кількість запитів за секунду) та наявність функції логування. З цими завданнями допоможе впоратися Modbus Poll або Modscan.

Слід враховувати, що зібрані логи потрібно буде аналізувати, тобто визначати кількість запитів, на які пристрій не відповів, виявляти збої, наприклад, спонтанна зміна даних в осередках і т. п. Для цього можна використовувати повноцінну SCADA-систему, або власноруч написану систему аналізу та візуалізації логів.

5.1.2 ModRSsim2

Програма ModRSsim2 – це симулятор протоколу Modbus для RS-232 та TCP/IP. Програма підтримує наступні реалізації протоколу:

– симуляція підлеглого пристрою з доступом за протоколом Modbus TCP/IP;

– симуляція підлеглого пристрою з доступом за протоколом Modbus RTU.

Середовище надає можливості тестування з використанням різних сценаріїв автоматизації.

Після завантаження програми ModRSsim2 необхідно провести початкові налаштування для роботи з протоколом Modbus. Для цього потрібно обрати відповідний тип протоколу в правому верхньому меню (рис. 5.3).

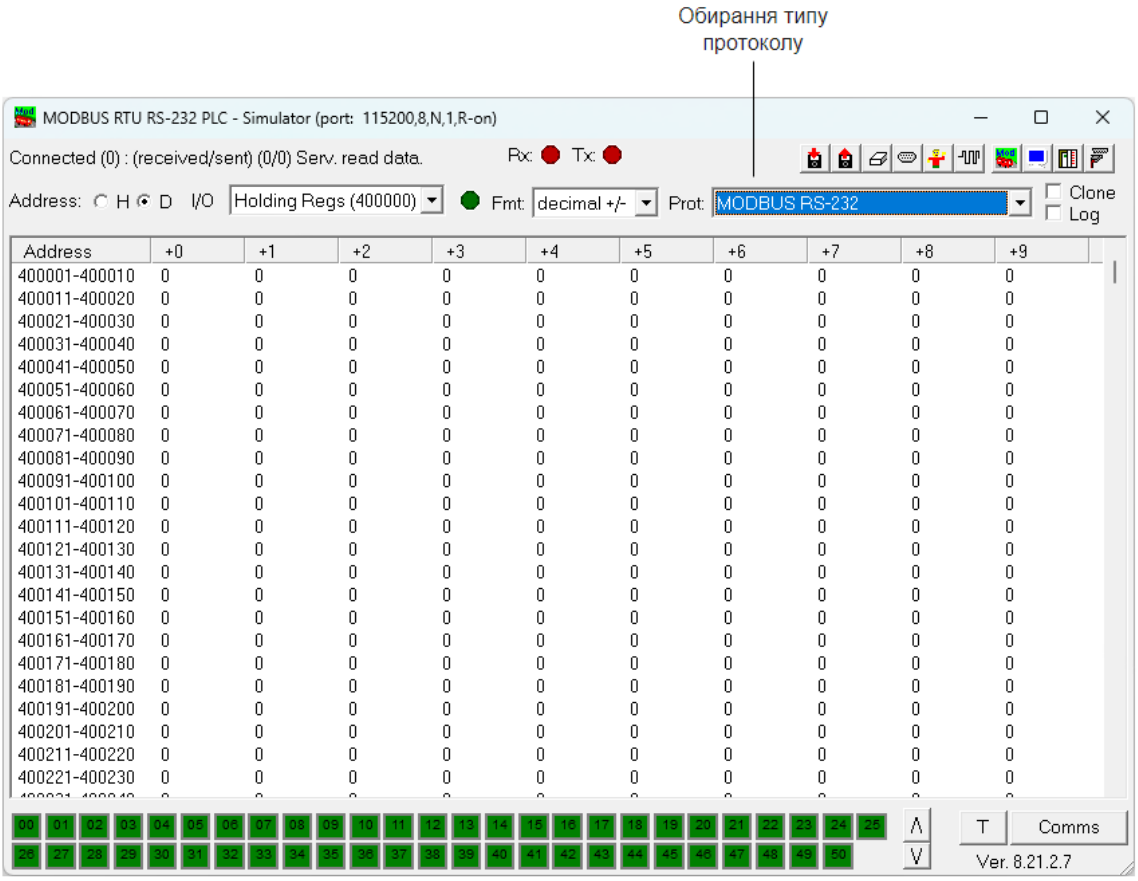


Рисунок 5.3 – Обирання типу протоколу

В меню, що випадає, можна обрати один з варіантів:

– Modbus RS232;

– Modbus TCP/IP;

– Allen Bradley DF1;

– Joy SCC DF1.

Приклад меню подано на рис. 5.4.

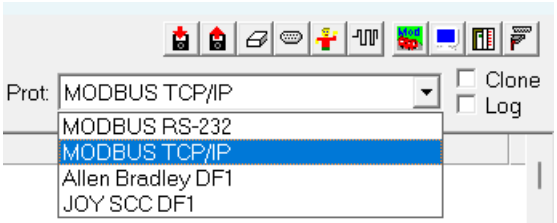


Рисунок 5.4 – Меню обирання типу протоколу

У випадку, якщо користувач вперше обирає протокол Modbus TCP/IP, система запропонує вікно налаштування мережного захисника Windows (рис. 5.5).

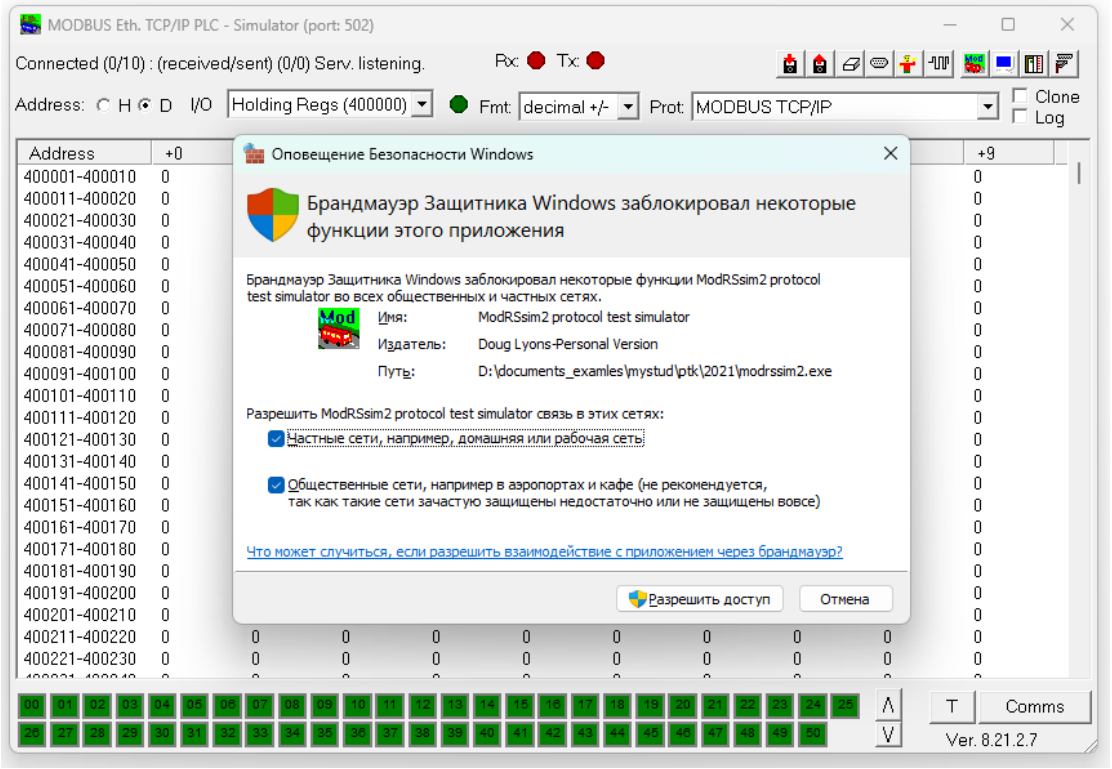


Рисунок 5.5 – Надання дозволу для роботи програми в робочій мережі

Після завершення налаштування мережного захисника необхідно обрати параметри з’єднання. Для цього в програмі передбачена відповідна кнопка (рис. 5.6)

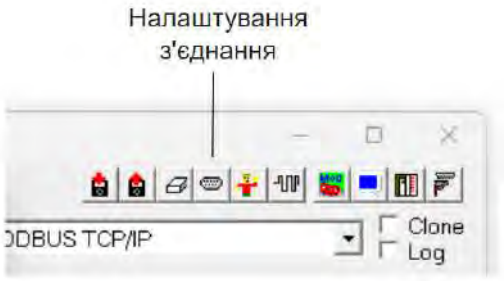


Рисунок 5.6 – Налаштування з’єднання

Параметри налаштування залежать від типу обраного протоколу. Якщо обрано TCP/IP, то в параметрах налаштування з’єднання необхідно перевірити правильність обирання порту, через який відбуватиметься передача даних (рис. 5.7). Також потрібно перевірити IP-адресу, яку прослуховує програма ModRSsim2. Зазвичай на локальному ПК обирається адреса 127.0.0.1 (localhost).

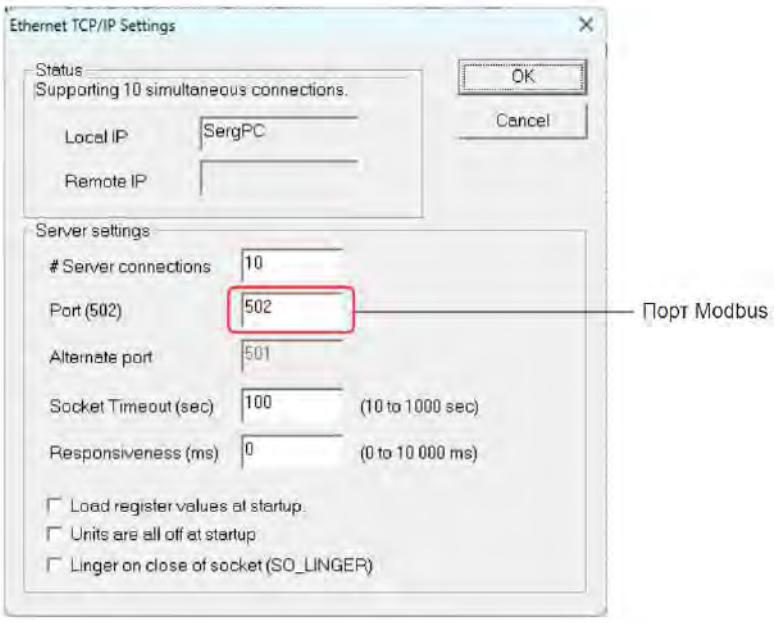


Рисунок 5.7 – Порт передачі даних для Modbus TCP/IP

У разі, якщо обрано тип протоколу Modbus RS-232, вікно налаштування буде мати вигляд, поданий на рис. 5.8.

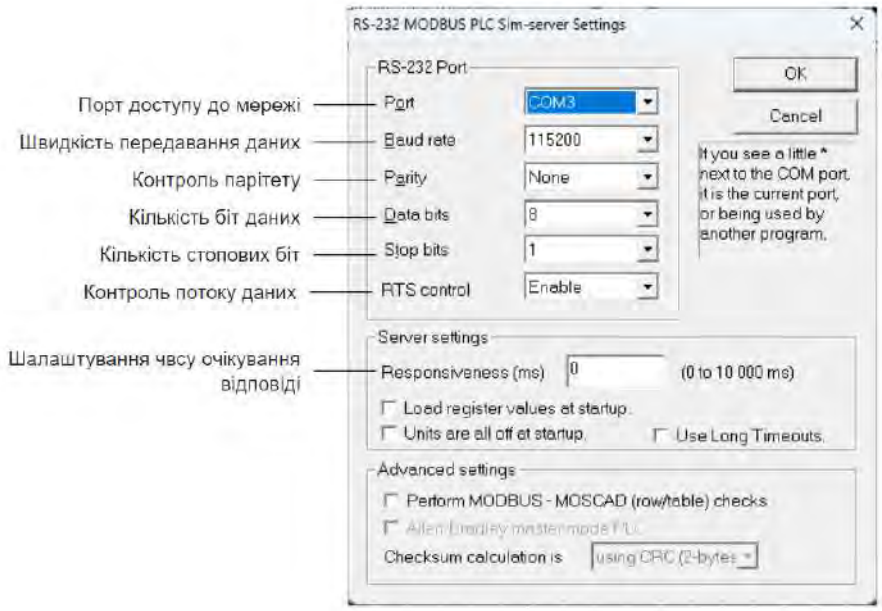


Рисунок 5.8 – Вікно налаштування параметрів з’єднання за протоколом Modbus RTU

В даному вікні налаштовуються:

– COM-порт для доступу до мережі Modbus;

– швидкість передачі даних;

– тип контролю правильності передачі даних;

– кількість біт даних в повідомленні (7 або 8);

– кількість стопових біт (1; 1,5; 2);

– тип управління потоком даних.

Також можна задати час очікування відповіді від підлеглого пристрою, якщо використовується нестабільна мережа, або виникли проблеми зі зв’язком.

Управління підключенням відбувається за допомогою відповідної кнопки (рис. 5.9).

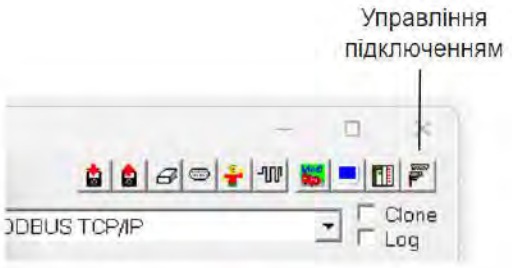
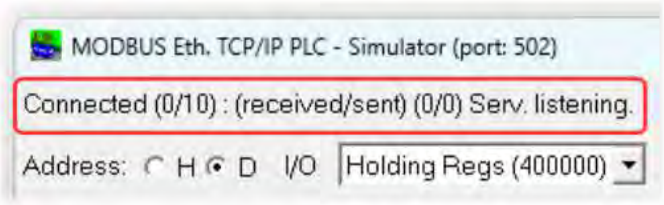
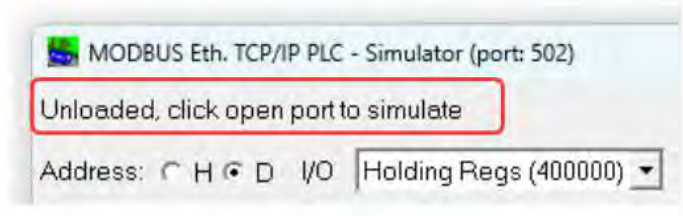


Рисунок 5.9 – Управління підключенням

Контролювати стан підключення можна за допомогою системного повідомлення, що виводиться на екран програми (рис. 5.10).



а)



б)

Рисунок 5.10 – Контроль стану підключення:
а) з’єднання встановлено; б) з’єднання відсутнє

5.1.3 Програмний симулятор «Майстер мережі Modbus TCP/IP»

Для налагодження пристроїв, що працюють під управлінням протоколу Modbus TCP/IP, а також для виконання лабораторних робіт присвячених вивченню мережних технологій, розроблено спеціальний програмний термінал. Інтерфейс програми подано на рис. 5.11.

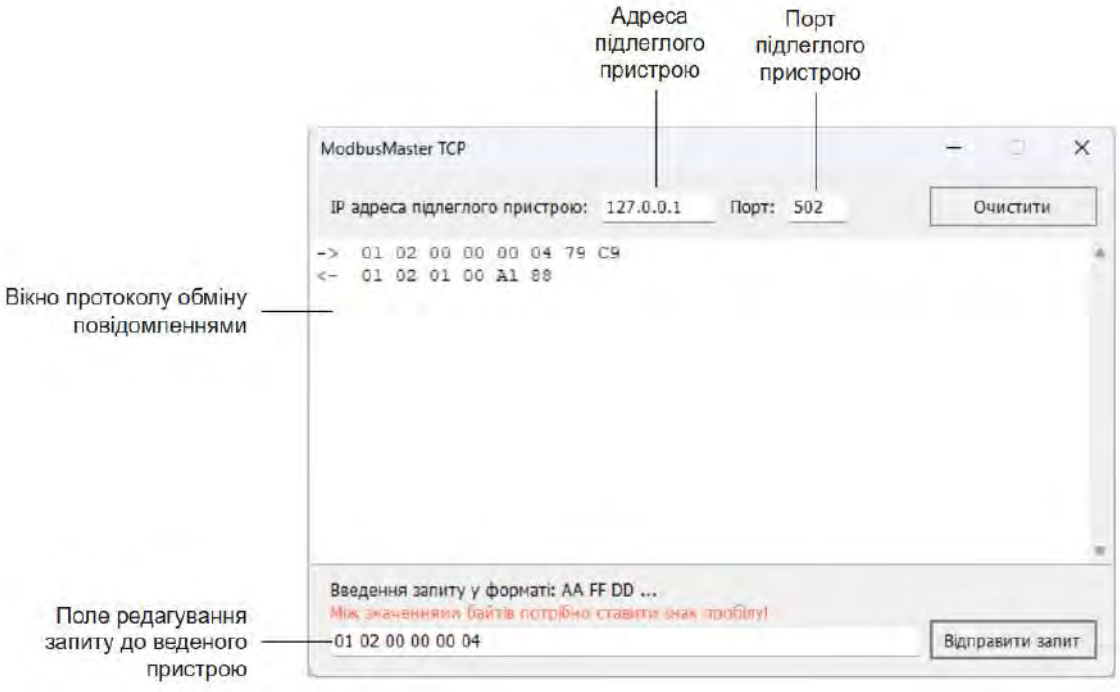


Рисунок 5.11 – Інтерфейс програми «Майстер мережі Modbus TCP/IP»

У верхній частині програми виконується налаштування підключення до мережі Ethernet. Тут можна ввести адресу підлеглого пристрою, до я кого будуть надсилатися запити, та вказати номер порту на підлеглому пристрою, через який відбувається прослуховування мережі.

За замовчуванням адреса підлеглого пристрою 127.0.0.1 (localhost), а порт доступу 502.

В нижній частині програми знаходиться поле введення запиту до підлеглого пристрою. Запит вводиться в HEX-форматі. Між байтами в запиті потрібно ставити пробіл. Наприклад, запит до підлеглого пристрою для читання дискретних входів матиме такий вигляд:

01 02 00 00 00 04

де 01 – адреса пристрою, але для протоколу Modbus TCP/IP вона ігнорується;

02 – номер функції;

00 00 – адреса першого дискретного входу;

00 04 – кількість входів для читання.

Контрольну суму CRC вводити не потрібно. Програма сама підраховує ці два байти та автоматично додає до запиту.

В середній частині інтерфейсу програми розташовано вікно для відображення історії переданих та отриманих пакетів. Переданим пакетам передує префікс «→», а прийнятим – «←» (рис. 5.12). Кнопка «Очистити» призначена для очищення історії повідомлень.

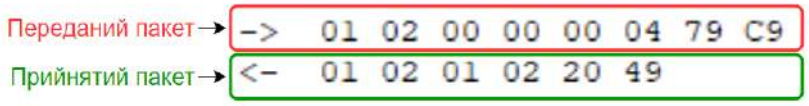


Рисунок 5.12 – Позначення переданих та прийнятих пакетів

5.1.4 Дослідження функції зміни стану вихідного контакту (0x05)

Перевіримо принцип увімкнення вихідних контактів на прикладі сумісного застосування програм симуляторів протоколу Modbus «ModRSsim2» та «Майстер мережі Modbus TCP/IP».

Після запуску програми ModRSsim2 виконаємо попередні налаштування:

– обираємо тип протоколу «Modbus TCP/IP»;

– виставляємо номер порту «502»;

– перевіряємо IP адресу, за якою відбувається прослуховування мережі (за замовчуванням – це 127.0.0.1);

– натискаємо на кнопку «Управління підключенням» (рис. 5.9) для поєднання з мережею та переходу в режим очкування команд від майстер-пристрою.

Наступним кроком є завантаження програми «Майстер мережі Modbus TCP/IP». Для цієї програми необхідно також виконати налаштування:

– в полі «IP адреса підлеглого пристрою» записати мережну адресу програми ModRSsim2 (за замовчуванням – це 127.0.0.1);

– в полі «Порт» записати той самий номер порту, що й в програмі ModRSsim2 (502).

Для дослідження роботи функції 0x05 «Зміна стану вихідного контакту» необхідно в полі «I/O» програми ModRSsim2 обрати адресний простір вихідних контактів (000000) (рис. 5.12).

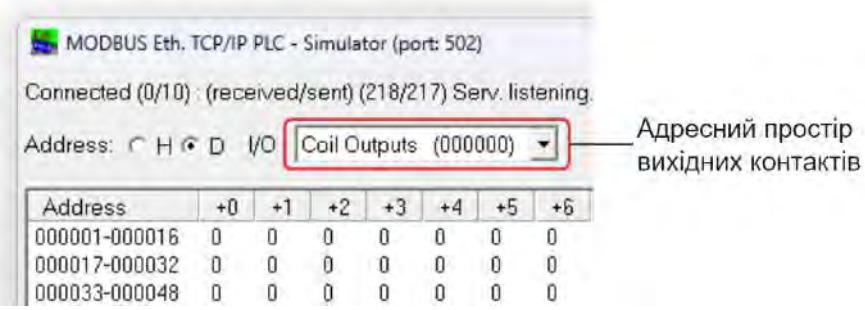


Рисунок 5.12 – Вибір адресний простір вихідних контактів

На початку роботи в програмі ModRSsim2 всі осередки, що відповідають вихідним контактам заповнені нулями (рис. 5.13). В програмі «Майстер мережі Modbus TCP/IP» поле обміну повідомлення також порожнє, а в полі введення запиту присутня підказка, яку потрібно видалити і записати новий запит, що відповідає функції 0x05.

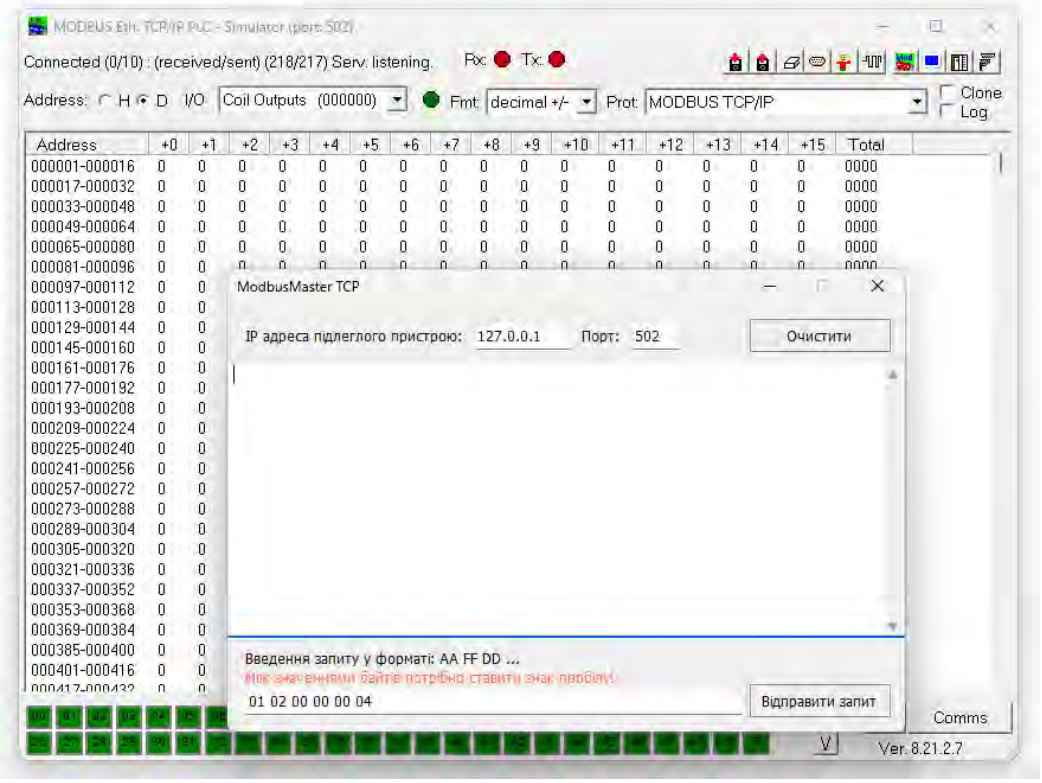


Рисунок 5.13 – Початковий режим роботи програмних симуляторів

Входячи з опису протоколу Modbus та функції 0x05 до підлеглого пристрою потрібно передати:

– адресу підлеглого пристрою;

– код функції;

– адресу дискретного виходу, Hi байт;

– адресу дискретного виходу, Lo байт;

– ознаку встановлення (0xFF 00), або скидання (0x00 00);

– контрольну суму CRC.

Для прикладу, спробуємо увімкнути вихідний контакт з адресою 000003 (рис. 5.14).

Address	+0	+1	+2	+3	+4	+5	+6	+7
000001-000016	0	0	0	0	0	0	0	0
000017-000032	0	0	0	0	0	0	0	0
000033-000048	0	0	0	0	0	0	0	0
000049-000064	0	0	0	0	0	0	0	0

Рисунок 5.14 – Поточний стан вихідного контакту з адресою 000003

Для увімкнення вихідного контакту з адресою 000003 створюємо такий запит:

Рисунок 5.15 – Запит до підлеглого пристрою для увімкнення вихідного контакту 0x0002

Тким чином, адреса записується в форматі, як подано на рис. 5.16.

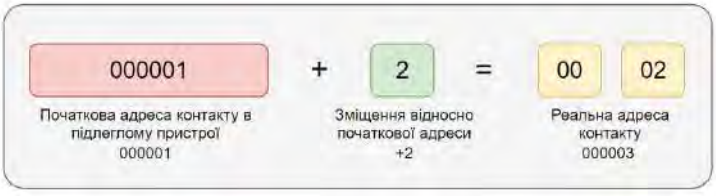


Рисунок 5.16 – Принцип формування адреси в програмі ModRSsim2

На рис. 5.17 подано приклад застосування побудованого запиту.

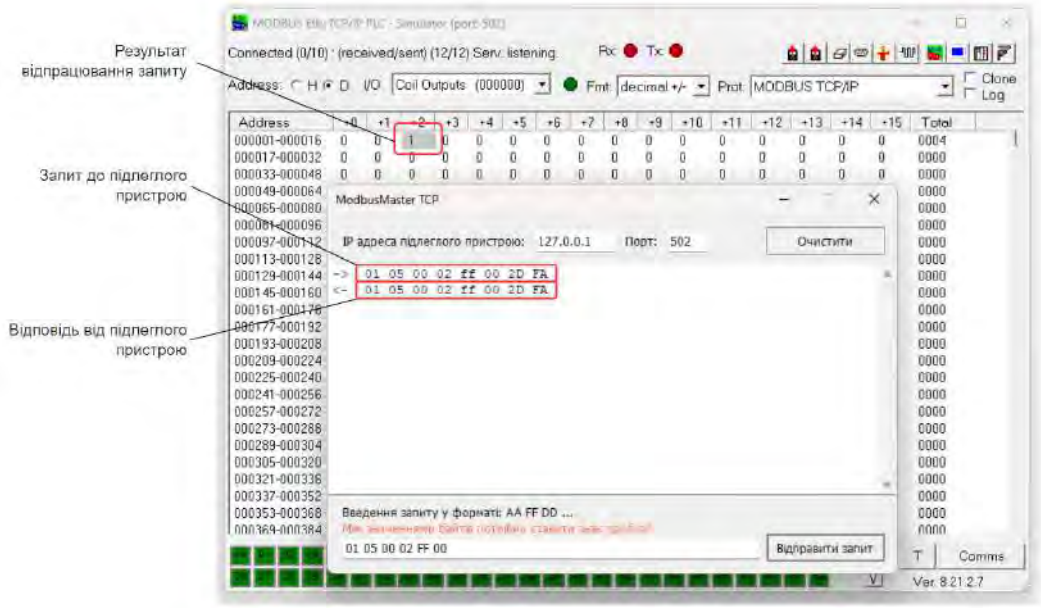


Рисунок 5.17 – Приклад застосування побудованого запиту

З поданого прикладу можна бачити:

– відправлений запит до підлеглого пристрою;

– відповідь від підлеглого пристрою (програма симулятор ModRSsim2);

– результат відпрацювання запиту.

Також можна бачити, що після вдалого виконання запиту, відповідь повністю повторює запит.

Для того, щоб віддалено перевірити правильність ввімкнення вихідних контактів потрібно застосувати функцію 0x01 (читання стану вихідних контактів). Відповідно до таблиці 4.3 в запиті необхідно передати:

– адресу підлеглого пристрою;

– код функції;

– адресу першого дискретного виходу, з якого починається читання, Hi байт;

– адресу першого дискретного виходу, з якого починається читання, Lo байт;

– кількість дискретних виходів для читання, Hi байт;

– кількість дискретних виходів для читання, Lo байт;

– контрольну суму CRC.

Запит для перевірки стану вихідного контакту з адресою 000003 має вигляд, поданий на рис. 5.18.

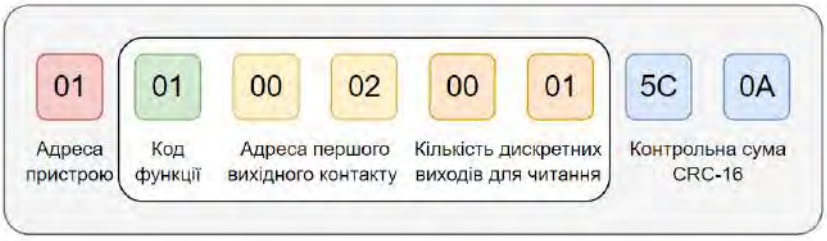


Рисунок 5.18 – Запит для перевірки стану вихідного контакту з адресою 000003

Відповідь на перевірочний запит до віддаленого пристрою має вигляд, поданий на рис. 5.19.

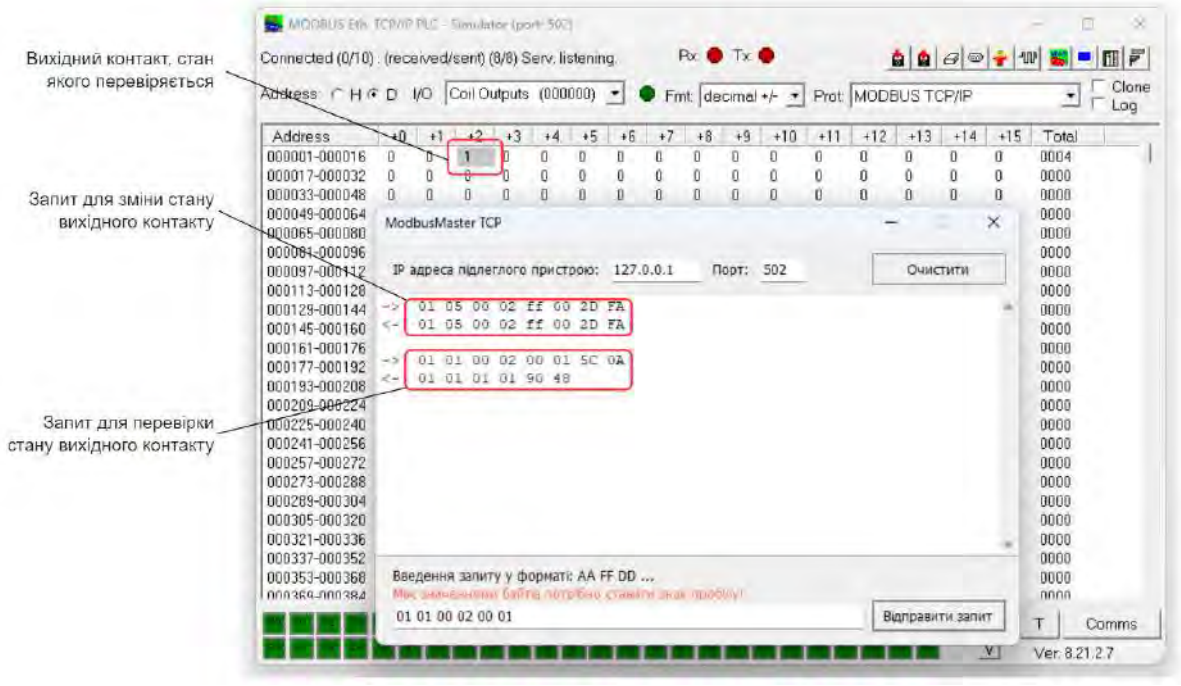


Рисунок 5.19 – Послідовність виконання запитів для зміни стану вихідного контакту та перевірки його поточного стану

Виходячи з рис. 5.19, отримана відповідь містить один байт даних, як подано на рис. 5.20.

Отриманий байт містить число 0x01, що відповідає очікуваному результату. Принцип формування даного байту відповіді підлеглим пристроєм подано на рис. 5.21.

Рисунок 5.20 – Отримана відповідь про стан вихідного контакту 000003

В запиті вказана кількість виходів, стан яких перевіряється, що дорівнює 1, тому в результаті ми можемо отримати тільки два варіанти байт даних: 0x00, або 0x01. В першому випадку вихідний контакт, що перевіряється вимкнено, в другому – увімкнено.

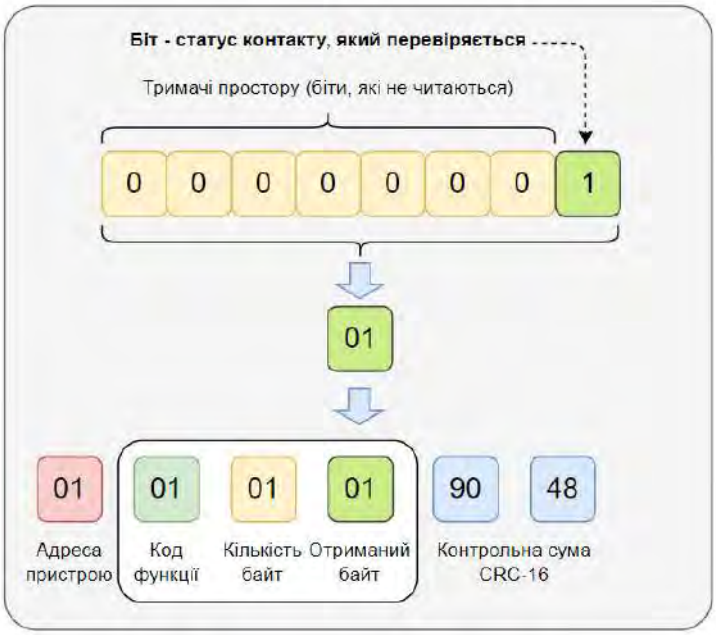


Рисунок 5.21 – Принцип формування відповіді підлеглим пристроєм

Виходячи з рис. 5.21, інші біти в підсумковому байті завжди будуть дорівнюватись нулю при перевірці лише одного вихідного контакту. Це так звані «тримачі простору».

Таким чином, запит для перевірки правильності увімкнення вихідного контакту 000003 показав, що він дійсно увімкнений та попередній запит підлеглим пристроєм виконано правильно.

5.1.5 Дослідження функції запису значення до регістрів зберігання (0x10)

За допомогою програми ModRSsim2 можна наочно дослідити правильність формування запиту для запису значення до регістрів зберігання від майстер-пристрою, який розробляється, або налаштовується.

В даному прикладі в якості майстер-пристрою також буде застосовано програмний симулятор «Майстер мережі Modbus TCP/IP».

Нехай за умовами задачі необхідно записати в регістри зберігання з адресами 400001, 400002 та 400003 числа 100, 200, 300, відповідно.

У відповідності до таблиці 4.11, для виконання поставленої задачі запит повинен містити:

– адресу підлеглого пристрою;

– код функції;

– адресу першого байту, Hi байт;

– адресу першого байту, Lo байт;

– кількість регістрів для запису, Hi байт;

– кількість регістрів для запису, Lo байт;

– кількість байт в полі даних;

– значення першого байту даних, Hi байт;

– значення першого байту даних, Lo байт;

– значення другого байту даних, Hi байт;

– значення другого байту даних, Lo байт;

– значення третього байту даних, Hi байт;

– значення третього байту даних, Lo байт;

– контрольну суму CRC.

В результаті формування запиту отримаємо таку структуру кадру, що зображена на рис. 5.22.

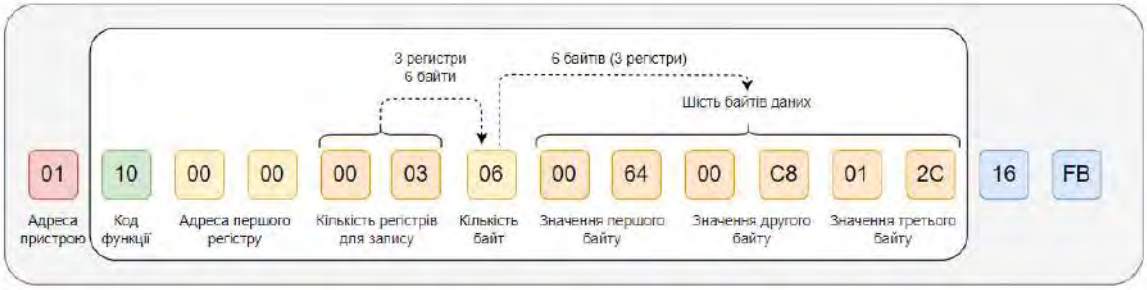


Рисунок 5.22 – Приклад запиту для запису значення в три регістри зберігання

Приклад виконання запиту подано на рис. 5.23. З поданого рисунку можна бачити, що вказані регістри 400001... 400003 змінили свій вміст та мають значення: 100, 200 та 300, відповідно.

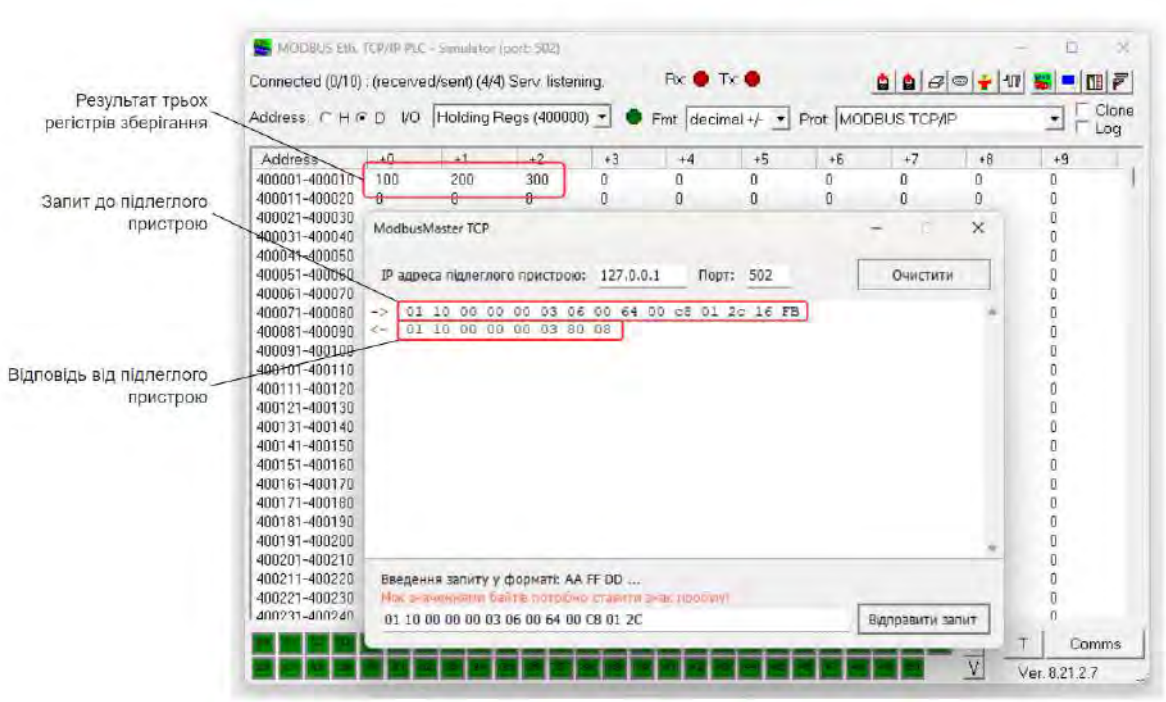


Рисунок 5.23 – Приклад виконання запиту для запису трьох регістрів зберігання

Відповідь від підлеглого пристрою підтверджує правильність виконання операції запису – у відповіді міститься інформація про запис трьох регістрів зберігання.

Якщо немає візуального контакту з підлеглим пристроєм, та необхідно перевірити стан певних регістрів зберігання, то формується новий запит з використанням функції читання декількох осередків даних за вказаними адресами.

У відповідності до таблиці 4.7 формуємо запит до підлеглого пристрою з метою одержання інформації про стан регістрів з адресами 400001-400003:

– адреса підлеглого пристрою (01);

– код функції (03);

– адреса першого байту, Hi байт (00);

– адреса першого байту, Lo байт (00);

– число регістрів для читання, Hi байт (00);

– число регістрів для читання, Lo байт (03);

– контрольна сума CRC (05);

– контрольна сума CRC (CB).

На рис. 5.24 подано приклад перевірки вмісту регістрів на підлеглому пристрої.

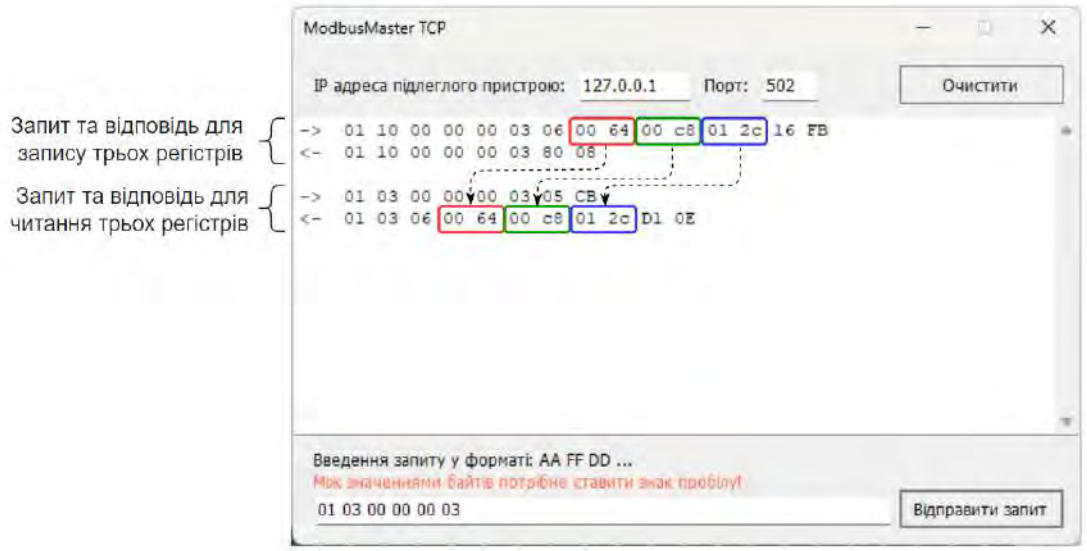


Рисунок 5.24 – Приклад перевірки вмісту регістрів на підлеглому пристрої

З поданого прикладу можна бачити, що передані значення трьох регістрів, під час виконання запиту запису, повторюються у відповіді від підлеглого пристрою при читанні.

Якщо розібрати відповідь, то у відповідності до таблиці 4.8, отримаємо:

– адреса підлеглого пристрою (01);

– код функції (03);

– кількість байт в полі даних (06);

– значення першого регістру (00 64);

– значення другого регістру (00 C8);

– значення третього регістру (01 2C);

– контрольна сума CRC (D1 0E).

Якщо перевести значення регістрів в десятирічну систему, то отримаємо ті ж самі значення: 0x0064 = 100, 0x00C8 = 200 та 0x012C = 300. Таким чином, можна зробити висновок, що операція запису та читання даних із регістрів зберігання спрацювала вірно.

5.2 Застосування протоколу Modbus для керування віртуальними приладами

5.2.1 Стислий опис макета світлової колони

Даний віртуальний прилад є цифровим двійником світлосигнальної колони для візуалізації стану промислового обладнання.

На рис. 5.25 подано зовнішній вигляд інтерфейсу віртуального приладу.

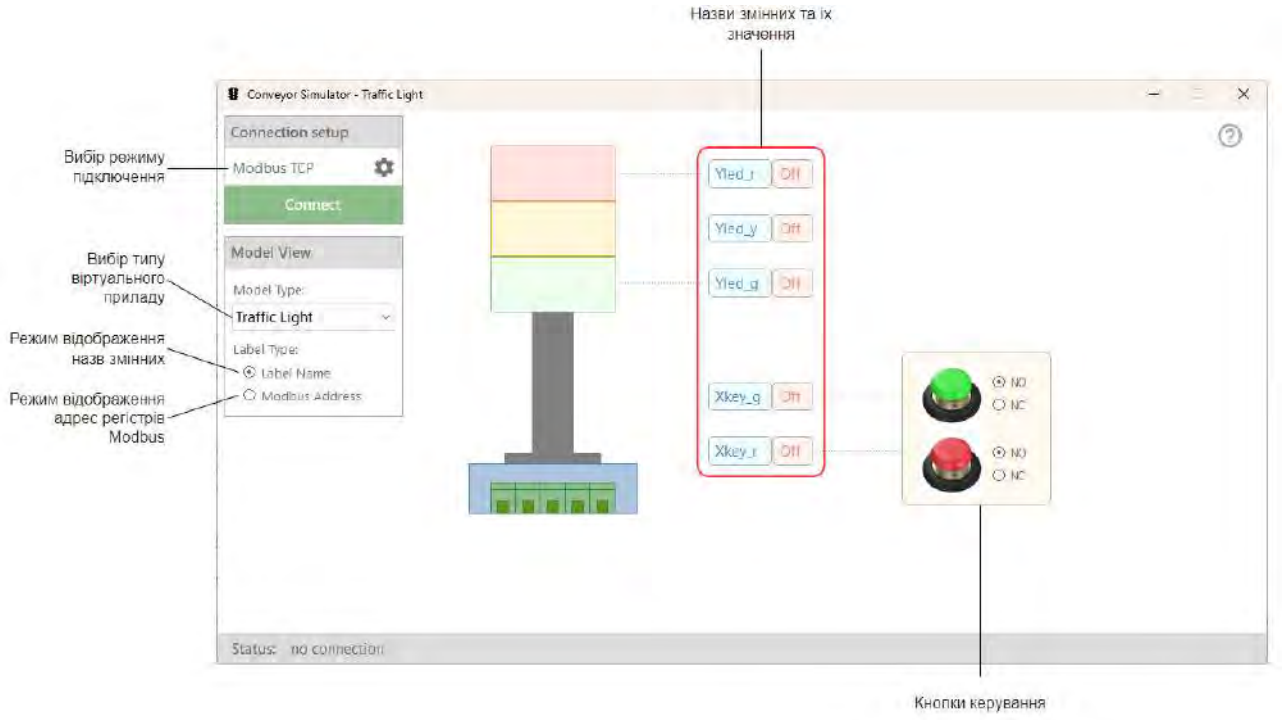


Рисунок 5.25 – Зовнішній вигляд інтерфейсу віртуального приладу «Світлова колона»

Віртуальний прилад повністю реалізує поведінку реального приладу та може працювати із зовнішньою програмою або іншим пристроєм за одним із можливих протоколів:

– Named Pipes для Inter-Process Communication;

– Modbus TCP/IP;

– Serial Protocol.

Для управління роботою даного приладу будемо використовувати протокол Modbus TCP/IP. Перед натисканням на кнопку «Connect» потрібно обрати саме цей тип протоколу у вікні налаштування параметрів підключення.

В інтерфейсі програми передбачено наявність візуалізації стану внутрішніх змінних, що відображають стан елементів віртуального макету. Внутрішні змінні мають назву для доступу до них з інших програм (наприклад LDmicro), або з віртуального ПЛК.

До змінних можна звернутись за їх адресою в пам’яті програми за допомогою інтерфейсу Modbus. На рис. 5.26 подано варіант інтерфейсу віртуального приладу при обиранні режиму відображення адрес регістрів Modbus.

Рисунок 5.26 – Інтерфейс віртуального приладу при обиранні режиму відображення адрес регістрів Modbus

Також, в даному режимі на екран виводиться довідник адрес регістрів Modbus та їх класифікація в залежності від типу.

Віртуальний прилад «Світлова колона» реалізує два типи регістрів:

– 1-бітні дискретні виходи (читання / запис) «Coils» (адресний простір 00001-10000);

– 1-бітні дискретні входи (читання) «Discrete Inputs» (адресний простір 10001-20000).

Кнопки керування можуть бути налаштовані, як пристрій з нормально відкритими контактами (Normal Open, або NO) та з нормально закритими контактами (Normal Close, або NC).

5.2.2 Приклад керування макетом світлової колони

Розглянемо приклад керування світловою колоною за допомогою програмного симулятора «Майстер мережі Modbus TCP/IP».

Наприклад, необхідно увімкнути жовтий сегмент колони. По-перше, необхідно дізнатися адресу вихідного контакту, що відповідає за цей сегмент у віртуальному приладі. Для цього перемикаємо інтерфейс у режим «Modbus Address» в секції «Model View». Навпроти жовтого сегменту знаходиться адреса «00002» та його поточне значення «Off» (вимкнено) (рис. 5.26).

Для того, щоб сегмент увімкнути, необхідно записати в регістр «00002» значення логічної одиниці. Сформуємо відповідний Modbus-запит, обравши функцію 0x05. Виходячи з опису протоколу Modbus та функції 0x05, до підлеглого пристрою потрібно передати:

– адресу підлеглого пристрою (01);

– код функції (05);

– адресу дискретного виходу, Hi байт (00);

– адресу дискретного виходу, Lo байт (01);

– ознаку встановлення (0xFF 00), або скидання (0x00 00);

– контрольну суму CRC.

Таким чином, запит до віртуального макету матиме наступний вигляд:

01 05 00 01 FF 00 CRC

Необхідно звернути увагу, що адреса в запиті вказується на одиницю менша, тому що за правилами протоколу Modbus рахування починається з 0 (00001 = 0x0000, 00002 = 0x0001, 00003 = 0x0002)

Перед перевіркою правильності побудови запиту виконуємо налаштування віртуального макету. Для цього викликаємо вікно конфігурації, натискаючи на кнопку з шестернею. На екрані ПК повинно з’явитися вікно налаштування параметрів підключення до мережі Modbus (рис. 5.27).

В даному вікні обов’язково потрібно перевірити IP-адресу, за якою буде звертатися майстер мережі до віртуального приладу. Зазвичай ця адреса відповідає IP-адресі того ПК, на якому завантажена ця програма, або це може бути стандартна адреса localhost (127.0.0.1), якщо і віртуальній прилад, і майстер мережі завантажені на одному ПК.

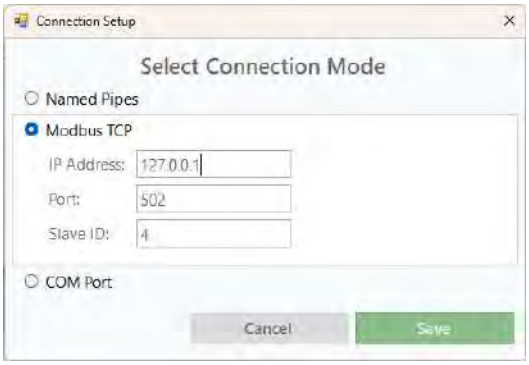


Рисунок 5.27 – Вікно налаштування параметрів підключення до мережі Modbus

Після перевірки IP-адреси необхідно натиснути на кнопку «Connect» для підключення до мережі Modbus TCP/IP та переходу в режим очікування команд від майстра мережі.

Результат виконання запиту подано на рис. 5.28.

Відповідь, що отримана від віртуального пристрою:

01 05 00 01 FF 00 DD FA

показує, що запит оброблено правильно і відповідний контакт увімкнуто.

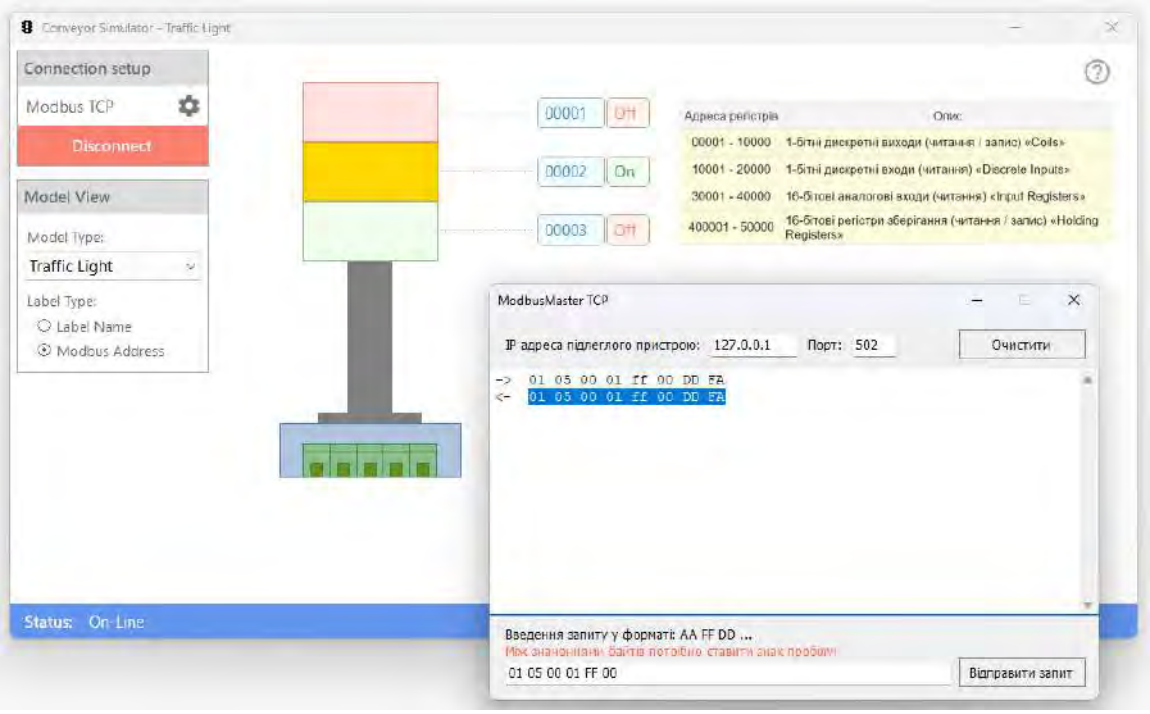


Рисунок 5.28 – Результат виконання запиту для увімкнення жовтого сегменту світлової колони

В наступному прикладі увімкнемо червоний та зелений сегменти колони, а жовтий сегмент вимкнемо.

Виконуючи аналіз задачі, бачимо, що нам потрібно змінити стан одночасно трьох різних регістрів: «00001» (червоний), «00002» (жовтий) та «00003» (зелений).

Для рішення даної задачі необхідно застосувати іншу функцію протоколу Modbus – 0x0F. Відповідно до стандарту протоколу необхідно сформувати запит у вигляді:

– адреса підлеглого пристрою (01);

– код функції (0F);

– початкова адреса дискретного виходу, Hi байт (00);

– початкова адреса дискретного виходу, Lo байт (00);

– кількість виходів, стани яких потрібно змінити, Hi байт (00);

– кількість виходів, стани яких потрібно змінити, Lo байт (03);

– кількість байт даних (01);

– перший байт даних (00000101 = 05);

– контрольна сума CRC.

Формування байту даних відбувається відповідно до рис. 5.29. Кожному сегменту світлової колони відповідає один біт. Розташування бітів відбувається справа наліво.

Рисунок 5.29 – Формування байту даних

Таким чином, запит до віртуального макету матиме наступний вигляд:

01 0F 00 00 00 03 01 05 CRC

Результат виконання запиту подано на рис. 5.30.

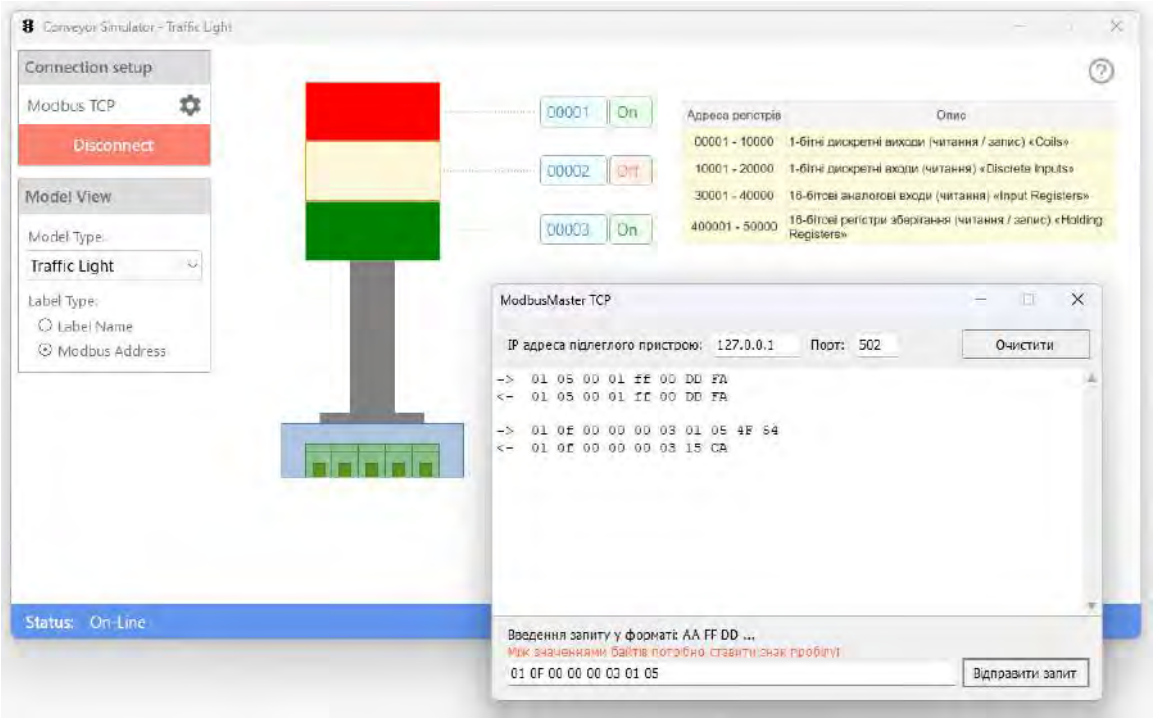


Рисунок 5.30 – Увімкнення сегментів 1 та 3 світлової колони

Як можна бачити з рис. 5.30, умови завдання виконані. Відповідні сегменти світлової колони змінили свій стан. Отримана відповідь, також підтверджує правильність виконання запиту.

5.2.3 Опис віртуального макету промислового конвеєра

Докладний опис віртуального макету промислового конвеєра наведено в [18]. На рис. 5.31 подано зовнішній вигляд інтерфейсу програми.

Конвеєр запускається записом логічної одиниці в змінну «YC_Moto». В цьому випадку починає рухатись полотно конвеєрної лінії, що візуально відображається на екрані ПК.

Видача деталей відбувається короткочасним записом логічної одиниці в змінну «YC_Stor1». Час утримання сигналу високого рівня має бути не менше 0,5 с, та не більше 2,5 с.

Видача деталей може здійснюватися автоматично. Для цього потрібно обрати режим «Auto» в списку, що випадає, який розташовано над змінною «YC_Stor1». В даному випадку зовнішні сигнали керування ігноруються.

Рисунок 5.31 – Зовнішній вигляд інтерфейсу віртуального макету промислового конвеєра

Наприкінці конвеєрної лінії розташовано приймач деталей. Деталі, що не були перенаправленні в процесі руху лінією, потрапляють в цей приймач та зберігаються там до моменту натискання на кнопку «Reset».

Кількість деталей, що міститься в магазині, та в кінцевому приймачі відображається у вигляді цифри на фоні деталі в відповідному пристрої. Наприклад, цифра «5», що розташована на червоній деталі, відповідає їх початковій кількості в магазині.

В процесі переміщення деталей конвеєрною лінією, їх напрям руху можна змінити за допомогою двох важелів «YC_Sp1» та «YC_Sp2». Управління важелями відбувається записом логічної одиниці в відповідну змінну. Увімкнений важіль візуально перекриває напрямок руху деталей та змушує її рухатись в один з двох накопичувачів – «XC_MSt1» або «XC_ MSt2». Кількість деталей в накопичувачах відображається цифрою на фоні останньої отриманої виробничої одиниці.

За допомогою запитів в форматі протоколу Modbus можна дізнатись про кількість деталей в магазині та у всіх накопичувачах. Відповідне значення записується в регістри зберігання. Розподіл адрес між регістрами зберігання та їх призначення подано в таблиці 5.1.

Таблиця 5.1 – Розподіл адрес між регістрами зберігання та їх призначення

Адреса регістру зберігання	Призначення регістру зберігання
40101	Кількість деталей в магазині
40102	Кількість деталей в накопичувачі №1
40103	Кількість деталей в накопичувачі №2
40104	Кількість деталей в кінцевому накопичувачі

Регістри зберігання не мають відповідних змінних, тому доступ до них можна здійснити тільки за допомогою протоколу Modbus. Вказані регістри призначенні тільки для читання. Запис в них значень не впливає на роботу конвеєра, тому що їх вміст оновлюється автоматично в процесі роботи програми керування віртуальним макетом.

В віртуальному макеті передбачено наявність трьох кнопок керування, функції яких визначає програміст під час створення технологічної програми.

Кожна кнопка пов’язана зі своєю змінною, тому змінна «XC_key1» відповідає верхній зеленій кнопці, змінна «XC_key2» відповідає середній червоній кнопці, а змінна «XC_key3» відповідає нижній червоній кнопці.

Також в макеті передбачені два сенсора. Перший сенсор «XC_sens1» встановлено на вході в конвеєр а другий «XC_sens2» – на виході. Перший сенсор крім дискретного виходу має ще три аналогових: «AC_R», «AC_G», «AC_B». Кожен з вказаних виходів пов’язано з внутрішнім 8-бітним АЦП. Значення цих трьох каналів дають змогу визначити колір деталі, що проходить повз перший сенсор.

Доступ до всіх змінних можна виконати за допомогою запитів за протоколом Modbus. Кнопки і сенсори організовані в блок вхідних контактів, виконавчі пристрої – в блок вихідних контактів, АЦП – у вхідні регістри.

Для зручності користування віртуальним макетом в програмі передбачено режим відображення Modbus-адрес замість реальних назв змінних (рис. 5.32).

В таблиці 5.2 подано відомості про відповідність адрес вихідних контактів в форматі Modbus до програмних змінних.

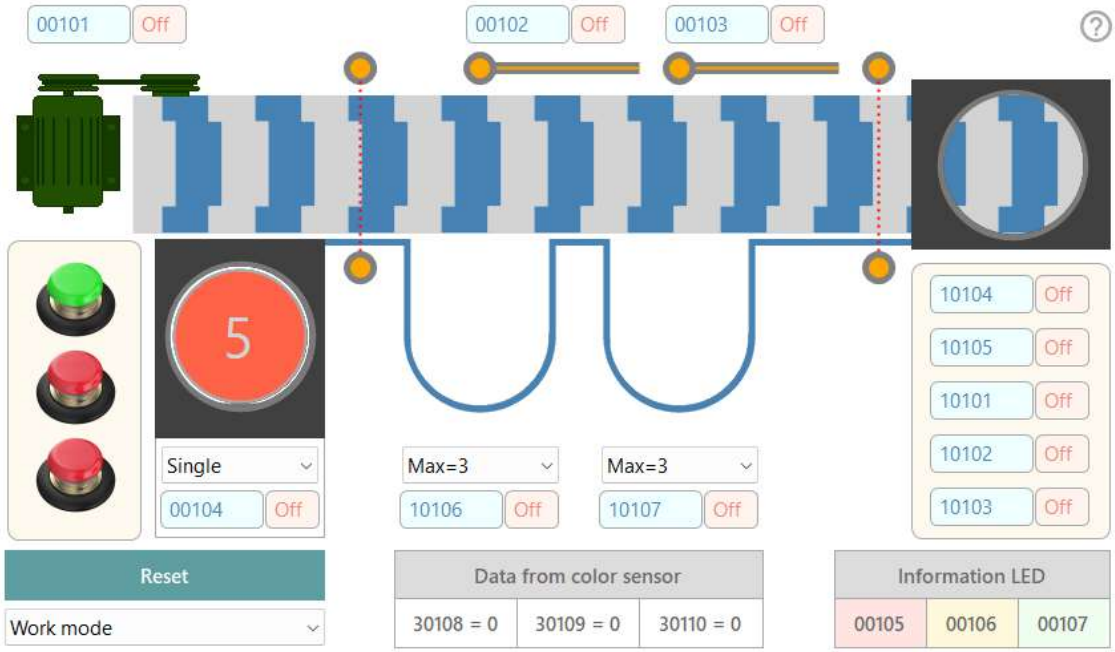


Рисунок 5.32 – Інтерфейс програми в режимі відображення Modbus-адрес

Таблиця 5.2 – Відповідність адрес вихідних контактів в форматі Modbus до програмних змінних

Адреса контакту	Назва змінної	Призначення контакту
00101	YC_Moto	Управління двигуном, що рухає конвеєрну стрічку
00102	YC_Sp1	Управління першим важелем розподілювача потоку деталей
00103	YC_Sp2	Управління другим важелем розподілювача потоку деталей
00104	YC_Stor1	Увімкнення або вимкнення приводу, що видає деталі з магазину
00105	YC_R	Контакт управління червоним сегментом інформаційного дисплею
00106	YC_Y	Контакт управління жовтим сегментом інформаційного дисплею
00107	YC_G	Контакт управління зеленим сегментом інформаційного дисплею

У всіх вихідних контактів активний рівень – логічна одиниця. Таким чином, поява логічної одиниці призводить до увімкнення відповідного виконавчого пристрою.

В таблиці 5.3 подано відомості про відповідність адрес вхідних контактів в форматі Modbus до програмних змінних.

Таблиця 5.3 – Відповідність адрес вхідних контактів в форматі Modbus до програмних змінних

Адреса контакту	Назва змінної	Призначення контакту
10101	XC_key1	Верхня зелена кнопка керування макетом
10102	XC_key2	Середня червона кнопка керування макетом
10103	XC_key3	Нижня червона кнопка керування макетом
10104	XC_Sens1	Дискретний вихід сенсора наявності деталі, що розташовано на вході конвеєрної лінії
10105	XC_Sens2	Дискретний вихід сенсора наявності деталі, що розташовано на виході з конвеєрної лінії
10106	XC_MSt1	Дискретний вихід сенсора, що сигналізує про досягнення максимальної кількості деталей в першому накопичувачі
10107	XC_MSt2	Дискретний вихід сенсора, що сигналізує про досягнення максимальної кількості деталей в другому накопичувачі

Спрацювання будь-якого датчика призводить до формування сигналу логічної одиниці на виході відповідного контакту.

В таблиці 5.4 подано відомості про відповідність Modbus-адрес каналам АЦП.

Таблиця 5.4 – Відповідність Modbus-адрес каналам АЦП

Адреса контакту	Назва змінної	Призначення контакту
30108	AC_R	Дані про колір деталі, R-канал
30109	AC_G	Дані про колір деталі, G-канал
30110	AC_B	Дані про колір деталі, B-канал

В макеті використовується 8-бітний АЦП. Враховуючи, що вхідні регістри 16-бітні, старший байт в відповіді завжди дорівнюватиме нулю.

5.2.4 Приклад керування макетом промислового конвеєра

Керування макетом буде відбуватись за наступним сценарієм:

– крок №1: увімкнення двигуна;

– крок №2: переведення першого важеля в положення «зачинено»;

– крок №3: видача деталі;

– крок №4: переведення першого важеля в положення «відчинено»;

– крок №5: видача деталі;

– крок №6: вимкнення двигуна конвеєра.

Виконаємо перший крок і увімкнемо двигун, що рухає конвеєрну стрічку.

Відповідно до таблиці 5.2 двигун вмикається з надходженням логічної одиниці на вихідний контакт з адресою 00101. Для управління єдиним вихідним контактом використовується функція 0x05.

Сформуємо команду протоколу Modbus з використанням функції 0x05 для передачі логічної одиниці за адресою 00101. Якщо перевести 00101 в шістнадцятирічний код, отримаємо 0x65 (рис. 5.33).

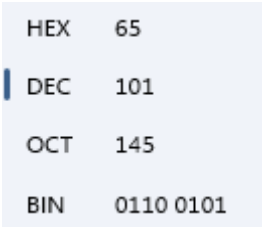


Рисунок 5.33 – Перетворення 00101 в шістнадцятирічний код

Під час використання протоколу Modbus всі адреси зменшуються на одиницю, тому в процесі формування запиту використовуватиме адресу вихідного контакту 0x64. Таким чином, остаточний запит матиме такий вигляд:

01 05 00 64 FF 00.

Після виконання даної команди конвеєрна стрічка повинна почати рухатися, а зображення двигуна змінитися та стати більш помітним на фоні програми. Також, навпроти мітки з адресою 00101 повинна з’явитися ознака «On» (рис. 5.34).

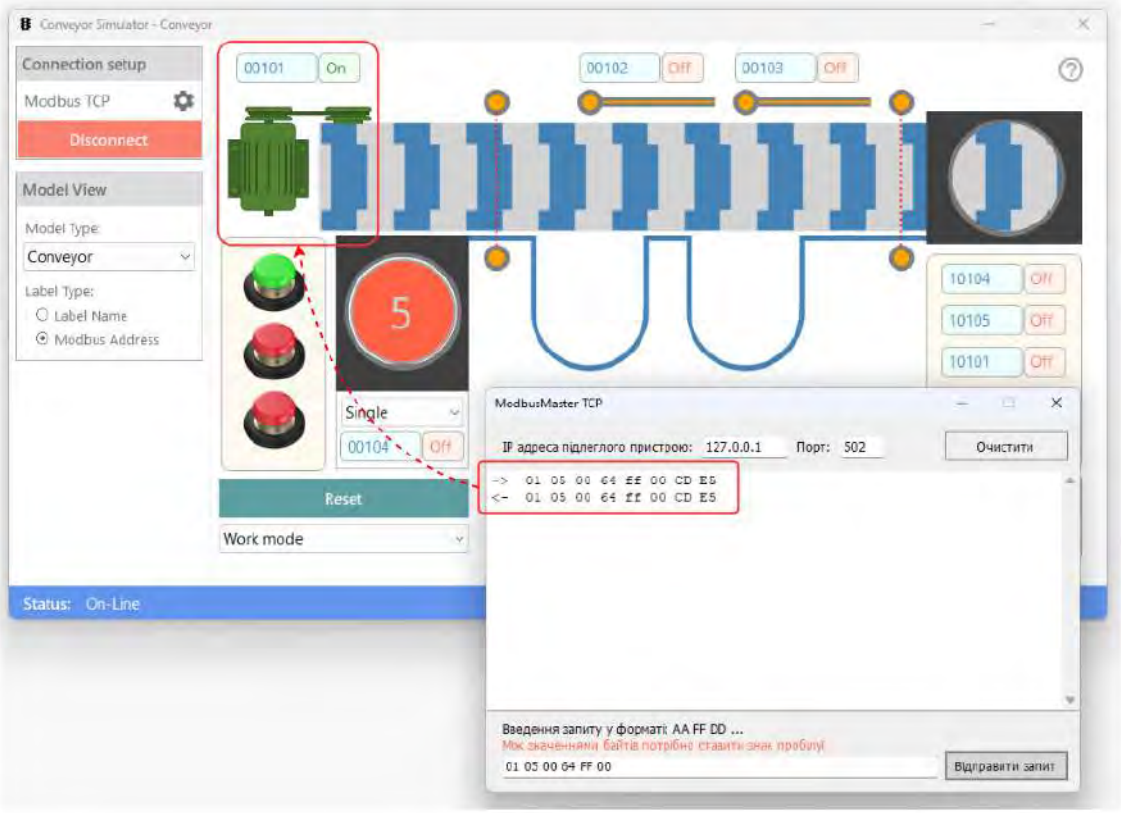


Рисунок 5.34 – Виконання кроку №1, увімкнення двигуна

Наступним кроком виконаємо переведення першого важеля в положення «зачинено». Для цього використовуємо також функцію 0x05 та записуємо до вихідного контакту з адресою 00102 логічну одиницю:

01 05 00 65 FF 00.

Результат виконання команди подано на рис. 5.35. Як можна бачити з рисунку, після виконання даної команди, перший важіль перекриває рух конвеєрної стрічки під кутом 45 градусів, а навпроти мітки з адресою 00102 з’явилася ознака «On».

Відповідь від підлеглого пристрою повністю ідентична до запиту. Це свідчить про правильність виконання запиту.

На третьому кроці необхідно виконати видачі однієї деталі з магазину. Магазин деталей має адресу 00104. Для видачі тільки однієї деталі за вказаною адресою, необхідно спочатку записати логічну одиницю та, не пізніше 2,5 с, записати логічний нуль.

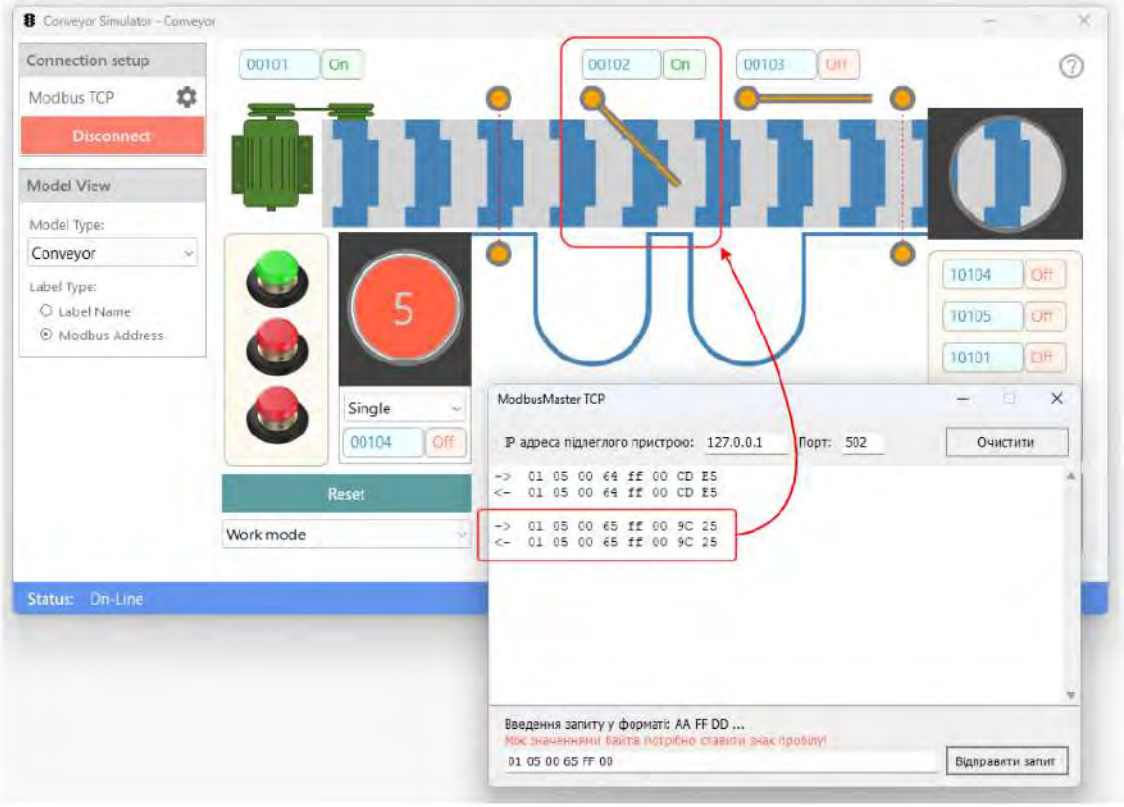


Рисунок 5.35 – Увімкнення першого важеля

Таким чином, спершу до віртуального макету буде передано команду:

01 05 00 67 FF 00,

а далі, не пізніше за 2,5 с, передається команда вимкнення штоку магазину:

01 05 00 67 00 00.

Результат виконання команди подано на рис. 5.36.

З даного рисунку видно, що деталь, рухаючись конвеєрною стрічкою, змінила свій напрям та потрапила до першого накопичувача. Кількість деталей в магазині зменшилася до 4, а в першому накопичувачі зросла до 1.

Четвертим кроком виконаємо повернення першого важеля в початкове положення «відчинено»:

01 05 00 65 00 00.

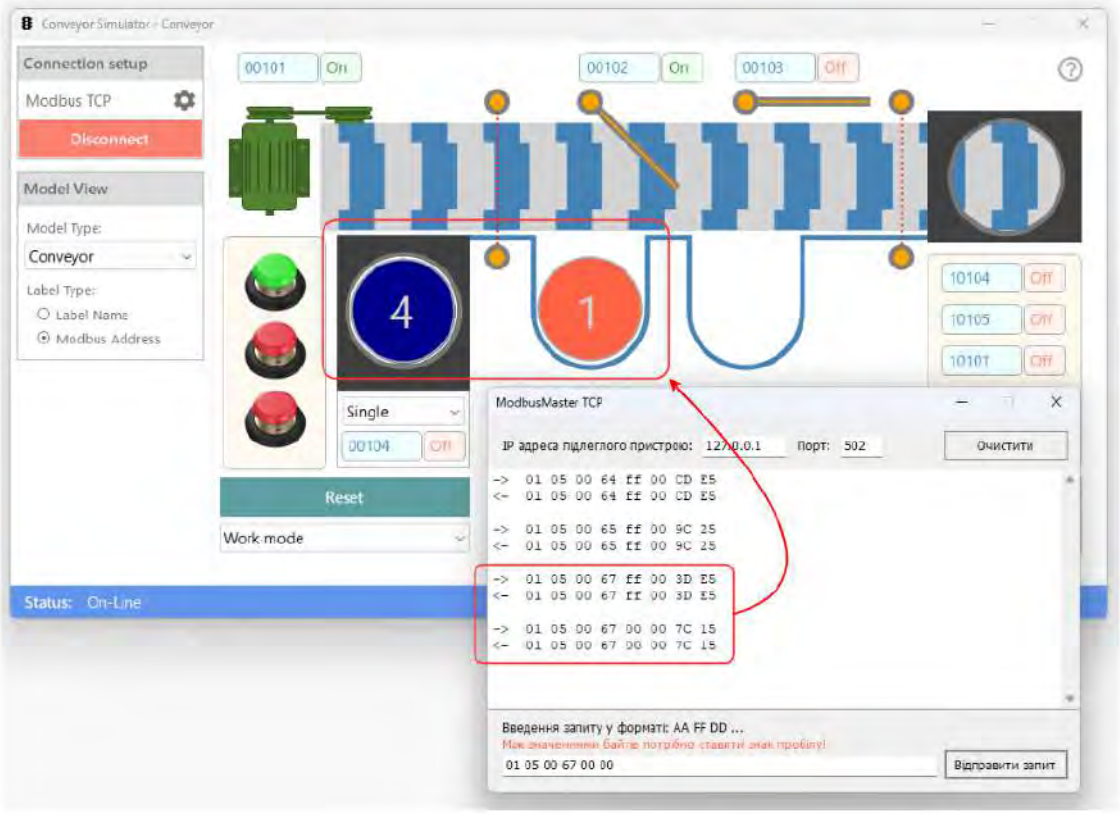


Рисунок 5.36 – Видача першої деталі

На п’ятому кроці виконаємо видачу ще однієї деталі на лінію за допомогою послідовності команд:

01 05 00 67 FF 00,
01 05 00 67 00 00.

На останньому кроці виконаємо зупинку конвеєра передавши команду:

01 05 00 64 00 00.

Результат виконання послідовності кроків 4-6 подано на рис. 5.37.

На даному етапі можна бачити, що в магазині та в накопичувачах знаходиться певна кількість деталей. Для того, щоб на віддаленому пристрої дізнатися про поточне розподілення деталей на виробничій лінії, потрібно виконати читання регістрів зберігання, що пов’язані з кожним із зазначених накопичувачів.

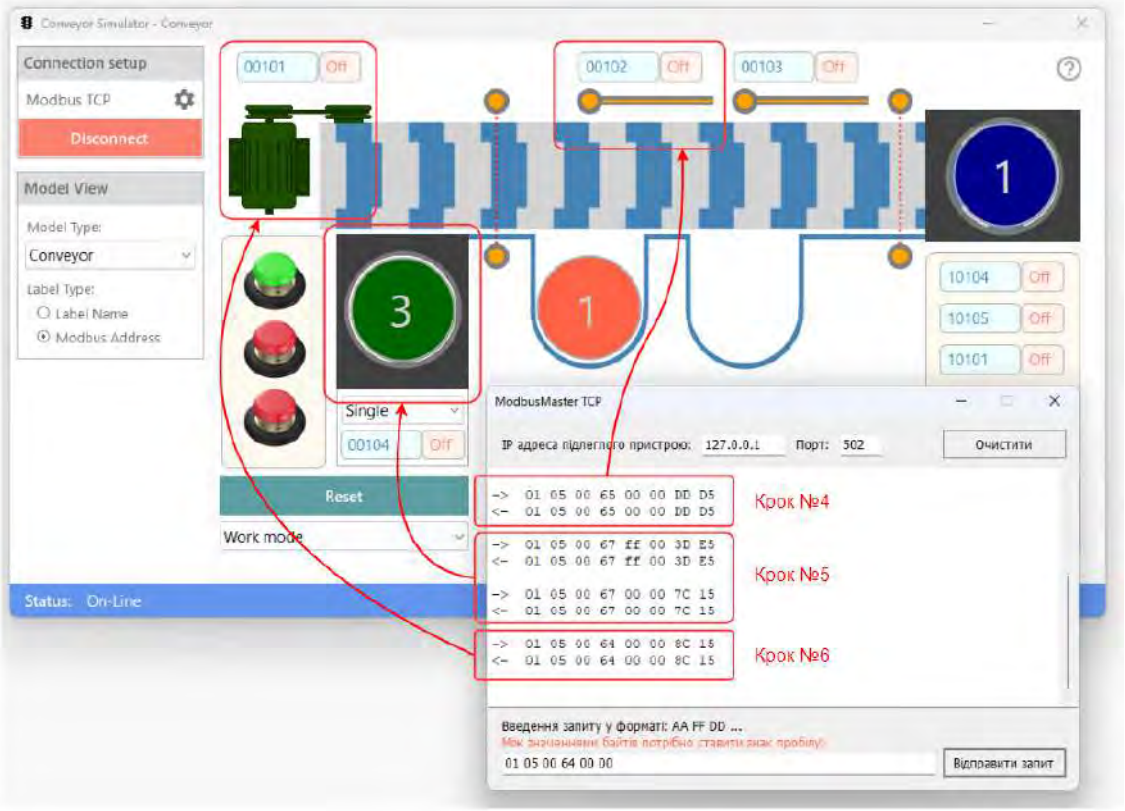


Рисунок 5.37 – Результат виконання послідовності кроків 4-6

Сформуємо команду для читання даних з чотирьох регістрів зберігання з адресами 40101-40104. Для цього використовуємо функцію Modbus 0x03. В запиті потрібно вказати:

– адресу підлеглого пристрою (01);

– код функції (03);

– адресу першого байту, Hi байт (00);

– адресу першого байту, Lo байт (64);

– число регістрів для читання, Hi байт (00);

– число регістрів для читання, Lo байт (04);

– контрольну суму CRC.

Таким чином, запит матиме наступний вигляд:

01 03 00 64 00 04.

Результат виконання запиту подано на рис. 5.38.

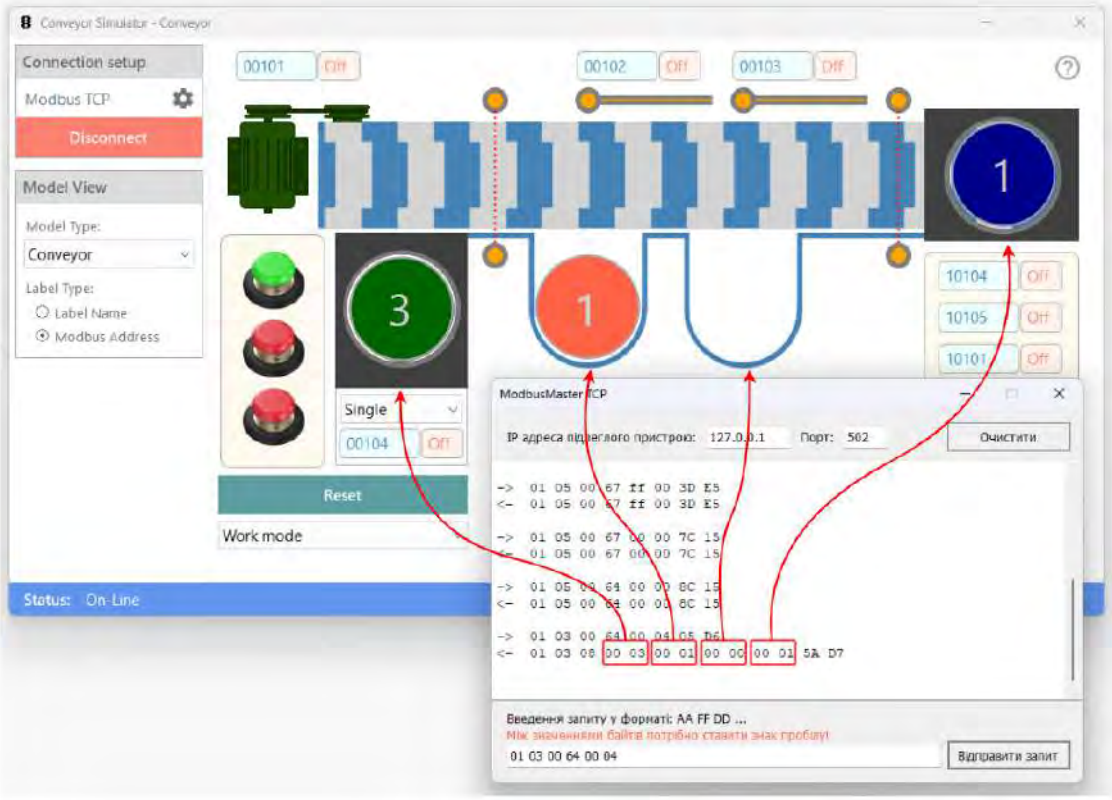


Рисунок 5.38 – Результат виконання запиту читання вмісту регістрів зберігання віртуального макету

Якщо розібрати отриману відповідь (01 03 08 00 03 00 01 00 00 00 01 5A D7), то у відповідності до таблиці розподілення адресів регістрів, можна побачити наступну інформацію:

– регістр 40101, що відповідає магазину деталей, містить значення 0x0003;

– регістр 40102, що відповідає першому накопичувачу, містить значення 0x0001;

– регістр 40103, що відповідає другому накопичувачу, містить значення 0x0000;

– регістр 40104, що відповідає кінцевому накопичувачу, містить значення 0x0001.

Таким чином, отримані дані повністю відповідають реальному розподіленню деталей на виробничій лінії (рис. 5.38).

5.3 Контрольні питання

1. Яке програмне забезпечення використовується для тестування пристроїв з підтримкою Modbus?
2. У чому полягає процес тестування пристроїв Modbus RTU під час розробки?
3. Які можливості надає програма ModRSsim2?
4. Для чого використовується симулятор «Майстер мережі Modbus TCP/IP»?
5. Як здійснюється дослідження функції зміни стану вихідного контакту (0x05) у середовищі тестування?
6. Як відбувається запис значень до регістрів зберігання за допомогою функції 0x10?
7. Які основні етапи налагодження мережі Modbus у лабораторних умовах?
8. Чим відрізняється тестування пристроїв у середовищах Modbus RTU та Modbus TCP?
9. Яке призначення віртуальних приладів у процесі вивчення протоколу Modbus?
10. Які основні компоненти входять до складу макета світлової колони?
11. Як здійснюється керування макетом світлової колони через Modbus-протокол?
12. Які функції використовуються для зміни кольору або стану світлової колони?
13. Що включає в себе віртуальний макет промислового конвеєра?
14. Як за допомогою протоколу Modbus реалізується керування конвеєром у віртуальному середовищі?
15. Які переваги надає використання віртуальних моделей для навчання роботі з протоколом Modbus?
16. Які протоколи підтримує Програма ModRSsim2?
17. Назвіть основні налаштування параметрів з’єднання за протоколом Modbus RTU.
18. Що таке контрольна сума?

6 ЗАСТОСУВАННЯ СЕРЕДОВИЩА CODESYS ДЛЯ РОЗРОБКИ ПРОГРАМ УПРАВЛІННЯ ТЕХНІЧНИМИ ЗАСОБАМИ АВТОМАТИЗАЦІЇ З ВИКОРИСТАННЯМ ПРОТОКОЛУ MODBUS

6.1 Ознайомлення з середовищем програмування CODESYS

6.1.1 Створення проєкту в CODESYS 3

Запуск програмного середовища CODESYS 3 може бути проведений з меню Пуск: Програми => 3S CODESYS => CODESYS => CODESYS V3.x, або подвійним клацанням по іконці на робочому столі. Результатом запуску буде вікно, подане на рис. 6.1.

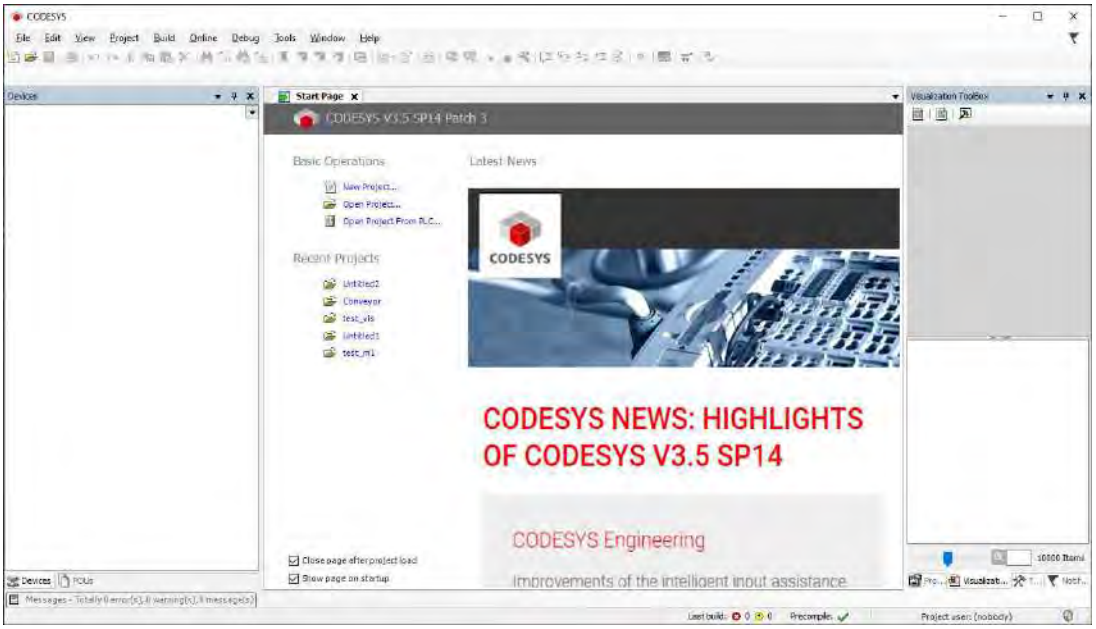


Рисунок 6.1 – Вікно CODESYS 3 після запуску

Для створення нового проєкту можна обрати пункт New project вкладки File, клацнути лівою кнопкою мишки по іконці на панелі швидкого виклику або скористатися поєднанням клавіш Ctrl + N. Результатом цих дій має стати поява вікна New Project, вид якого подано на рис. 6.2. У цьому вікні потрібно обрати пункт Standard project категорії General. У розділі Name можна дати ім'я проєкту, а в розділі Location – шлях його збереження на ПК.

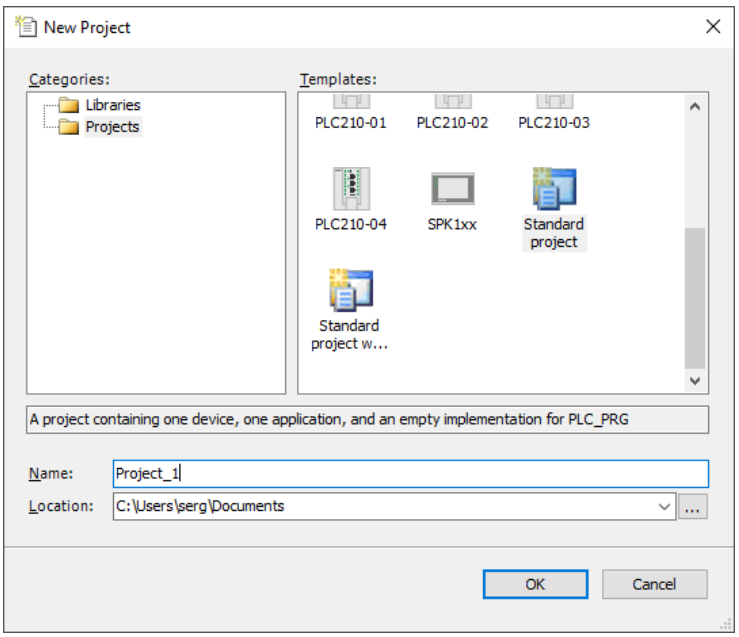


Рисунок 6.2 – Створення нового проєкту і визначення адреси його збереження

Після збереження проєкту з'являється вікно вибору виконавчого пристрою і мови основної програми. Його вигляд подано на рис. 6.3.

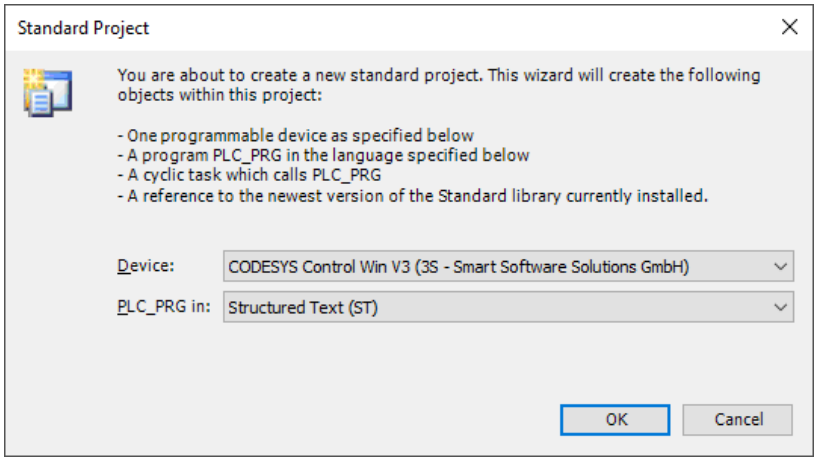


Рисунок 6.3 – Вибір пристрою і тексту основної програми

Як виконавчий пристрій у цьому прикладі буде використаний внутрішній віртуальний ПЛК середовища CODESYS 3. Цей пристрій реалізує повнофункціональний ПЛК на ПК, дозволяючи працювати в тому числі і з зовнішніми пристроями, підключеними до COM-портів комп'ютера. Для його використання в пункті Device виберемо CODESYS SP Win V3.

Мовою проєкту PLC_PRG обираємо мову структурованого тексту ST. Після задання цих параметрів дерево проєкту набуде вигляду, поданого на рис. 6.4.

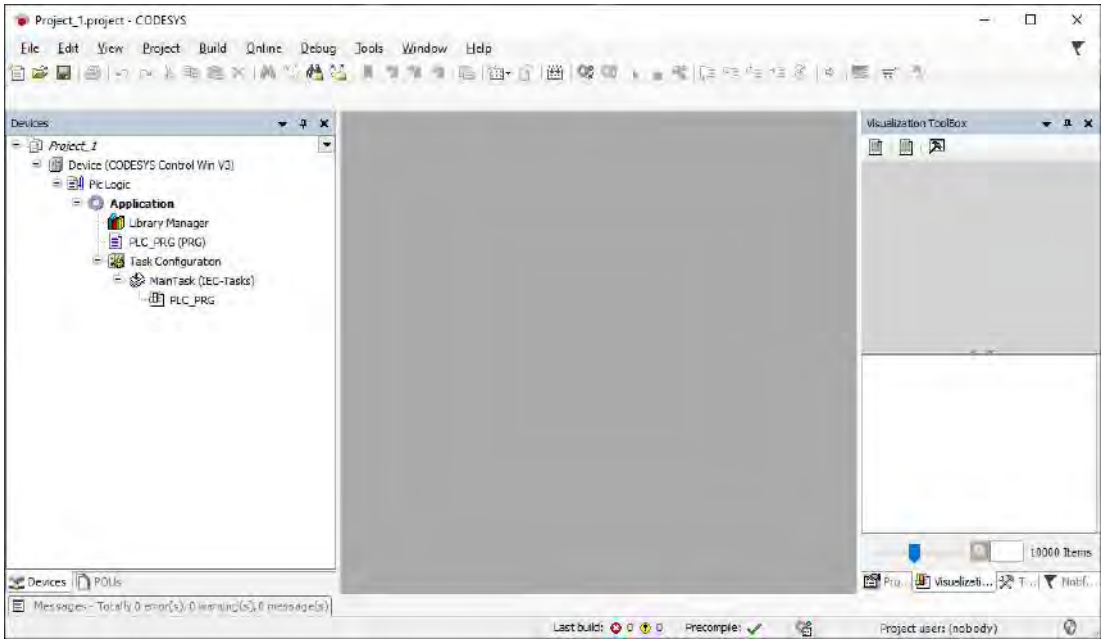


Рисунок 6.4 – Дерево проєкту

Головне вікно містить такі основні частини проєкту:

– вкладки Devices (дерево пристроїв) і POU (функції, функціональні блоки і програми проєкту) в лівій частині вікна екрану;

– робочу область (на рис. 6.4 неактивна, пофарбована сірим кольором, розташована у правій частині екрану);

– Messages (рядок повідомлень), що показує число і статус службових повідомлень, розташований у правій центральній частині екрану;

– Description, що містить опис етапів роботи CODESYS і розшифрування службових повідомлень і розташована в нижньому правому куті екрана.

У дереві проєкту з'явилася його назва (Oven2), вид пристрою (CODESYS Control Win V3) та наявні програми (Application), включаючи менеджер завдань і бібліотек, а також основну програму PLC_PRG.

Зовнішній вигляд менеджера бібліотек у конфігурації за замовчуванням подано на рис. 6.5. Перелік містить бібліотеку Standard, що включає лічильники, таймери, перемикачі та інші основні функціональні блоки.

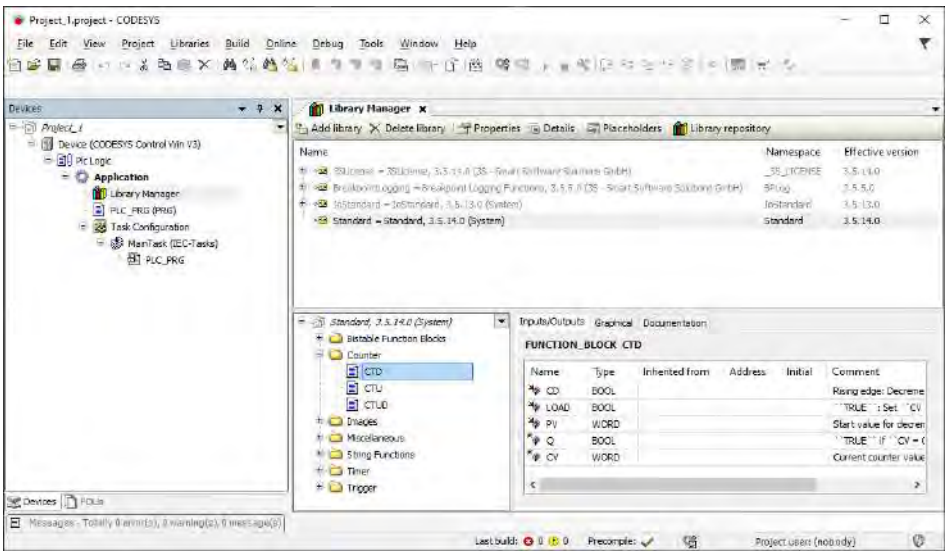


Рисунок 6.5 – Стандартна бібліотека CODESYS 3 (додається за замовчуванням)

Для прикладу створимо найпростіший проєкт з двома змінними *x* і *y* як подано на рис. 6.6. У цьому проєкті змінна *x* збільшується на 1 кожен цикл роботи ПЛК, а змінна *y* аналогічно зменшується.

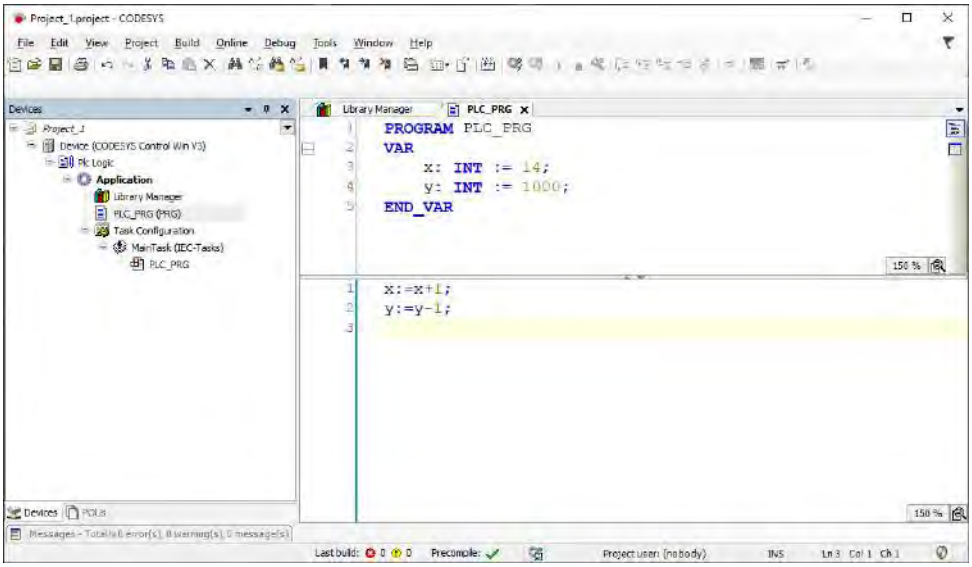


Рисунок 6.6 – Вигляд програми інкремента / декремента

Задання типів і початкових значень змінних може бути здійснено в розділі змінних PLC_PRG (між службовими словами VAR і END_VAR) або за допомогою вікна автооголошення, яке викликається натисканням кнопки F2 в редакторі вихідного коду програми, вид якого подано на рис. 6.7.

Після створення програми виконується її компіляція за допомогою виклику опції головного меню або після натискання на кнопку F11.

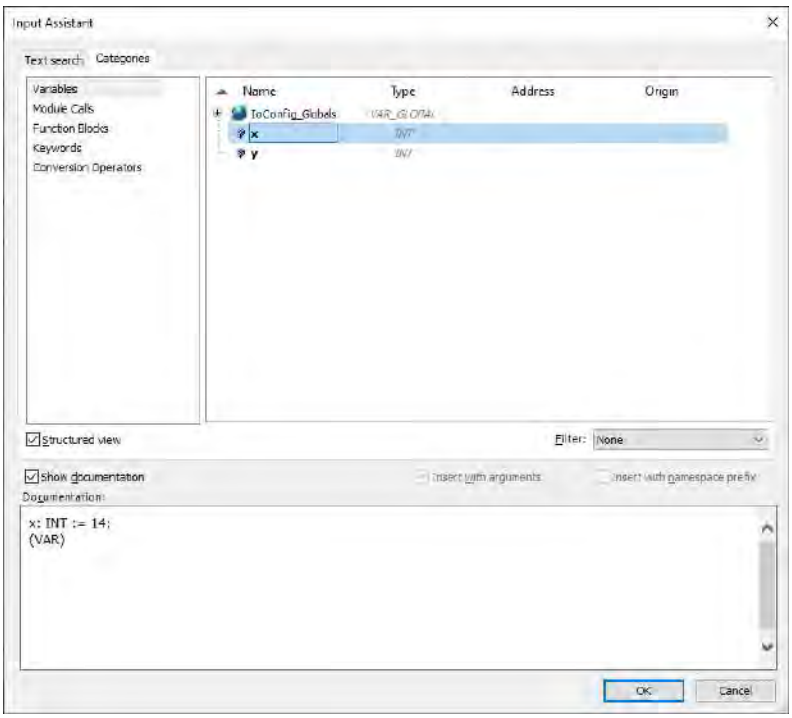


Рисунок 6.7 – Вікно асистента введення змінної

6.1.2 Робота в режимі емуляції

Для перевірки працездатності проєкту без підключення до контролера може бути використаний режим емуляції. У ньому можна провести налаштування і моделювання роботи програми без завантаження в ПЛК.

Вибір режиму роботи «Емуляція» виконується встановленням позначки в пункті Online / Simulation, як подано на рис. 6.8.

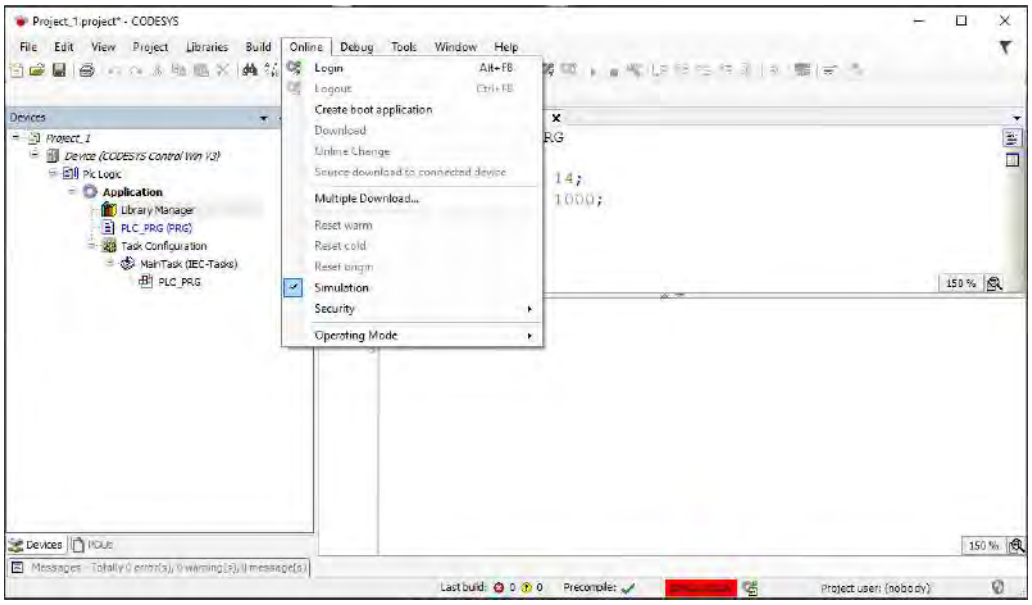


Рисунок 6.8 – Вибір режиму емуляції

Після вибору цього пункту із запуском програми на виконання в меню статусу з'являється напис «ЕМУЛЯЦІЯ». Запуск програми здійснюється двома командами: логін (Alt + F8) або старт (F5). Після запуску програма виконує свій код. Біля кожної змінної відображається її поточне значення, як подано на рис. 6.9.

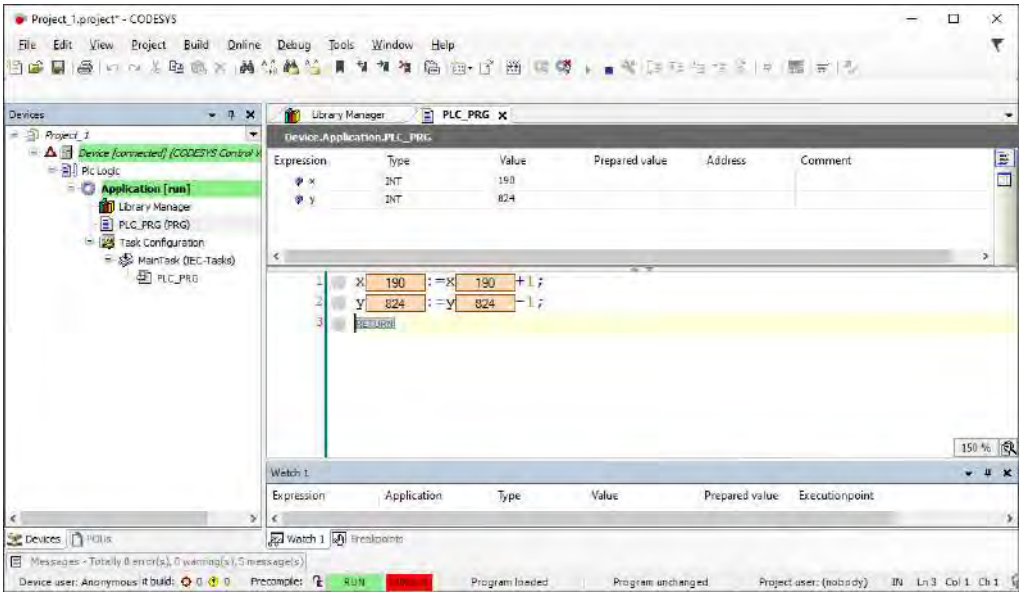


Рисунок 6.9 – Робота програми в режимі емуляції

Примусово змінити значення змінної можна в таблиці змінних у полі Prepared Value (Підготовлене значення) (рис. 6.10).

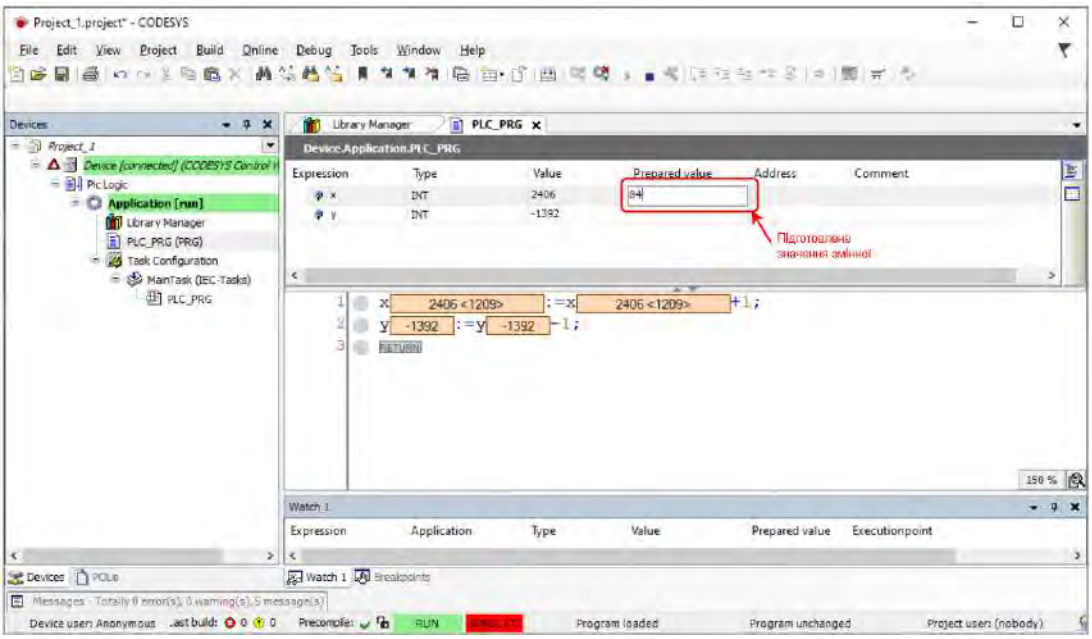


Рисунок 6.10 – Змінення значення змінної в проєкті

Також змінити значення змінної можна, викликавши вікно Prepare Value подвійним клацанням лівої кнопки мишки на необхідній змінній (рис. 6.11).

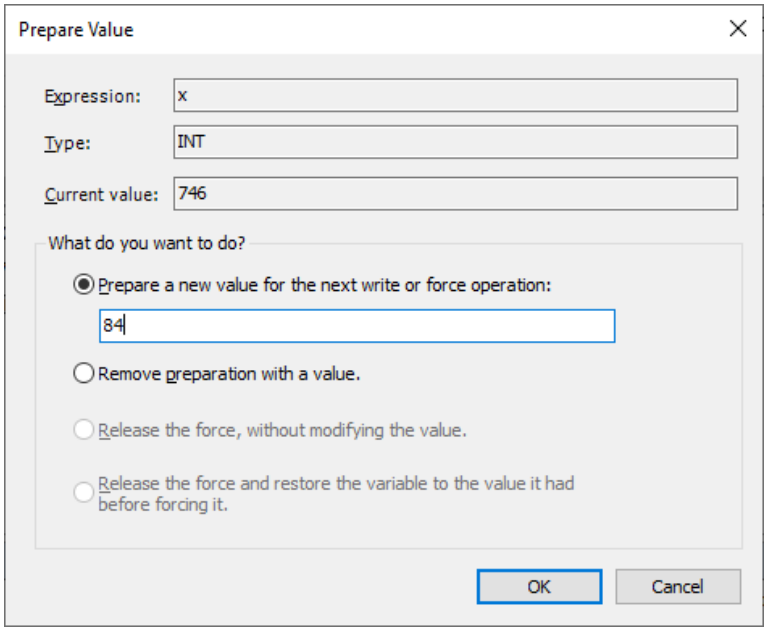


Рисунок 6.11 – Вікно змінення значення змінної

Встановлені значення з'являться в кутових дужках (< >) праворуч від дійсного значення змінної. Встановити змінені значення змінних можна вибором пункту меню Debug / Write values або сполученням клавіш Ctrl + F7.

6.1.3 Робота з віртуальним контролером CODESYS 3

Можливості користувача під час роботи в режимі симуляції обмежені. Розширити їх без підключення реального ПЛК дозволяє використання віртуального контролера CODESYS 3. Він встановлюється разом з програмним середовищем CODESYS 3 і запускається за допомогою Gateway-сервера.

Gateway-сервер автоматично запускається як сервіс із запуском системи. Переконайтеся, що на панелі задач є кольорова іконка, яка вказує на те, що сервер запущено. На рис. 6.12 показано зовнішній вигляд іконки в різних станах серверу. Сіра іконка свідчить про те, що gateway на даний момент зупинений. Ця іконка є частиною програми GatewaySysTray, яка призначена для контролю і спостереження за сервісом Gateway. Вона містить меню, яке має команди start і stop, що дозволяє користувачеві зупиняти і перезапускати сервіс вручну.

Меню також містить команду Exit Gateway Control, яка закриває тільки програму GatewaySysTray, але не сервіс Gateway. Програма GatewaySysTray

запускається автоматично із запуском Windows. Проте її можна також запустити з меню Програми.

Сервер запущений | Сервер зупинений

Рисунок 6.12 – Зовнішній вигляд іконки в різних станах Gateway-серверу

ПЛК (CODESYS SP Win V3) доступний як сервіс після запуску системи. На панелі задач він позначений іконкою (рис. 6.13) з різним зафарбленням для стану «запущений» і для стану «зупинений».

ПЛК-сервіс запущений | ПЛК-сервіс зупинений

Рисунок 6.13 – Стан віртуального ПЛК

ПЛК-сервіс може автоматично запускатися під час запуску системи, якщо це підтримується самою системою. В іншому випадку для його запуску необхідно вручну застосувати команду «Start PLC» з меню, яке відкривається клацанням мишки по іконці.

Для підключення віртуального контролера в дереві проєкту подвійним клацанням по пристрою Device (CODESYS SP Win V3) відкриємо діалог PLCWinNT з вкладкою Communication settings. Якщо встановлюється з'єднання в CODESYS V3.x в перший раз, то спочатку вам необхідно поставити локальний Gateway-сервер. Для цього використовується кнопка Add gateway, після чого з'явиться вікно, вигляд якого подано на рис. 6.14.

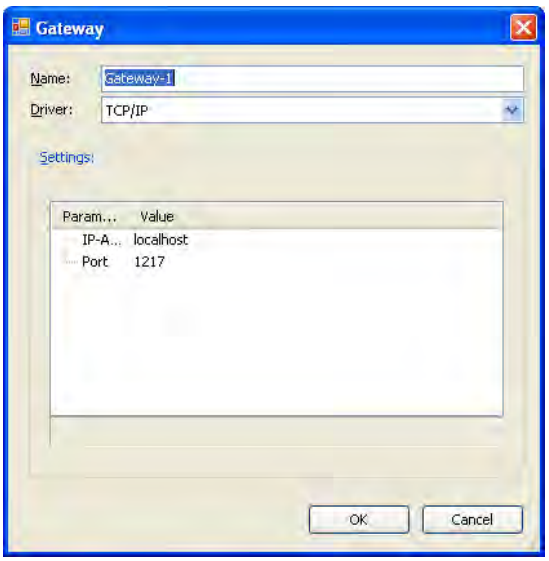


Рисунок 6.14 – Вікно додавання каналу Gateway

Якщо під час попередніх сесій вже налаштовувався сервер, він буде відображений у діалозі параметрів з'єднання, як подано на рис. 6.15. У такому випадку можна пропустити цей крок і перейти відразу до установки каналу зв'язку з пристроєм.

Рисунок 6.15 – Діалогове вікно параметрів з'єднання

У діалоговому вікні необхідно натиснути кнопку Scan network для отримання списку доступних мережних пристроїв, як це показано на рис. 6.16. У разі невдалого підключення необхідно перевірити налаштування цільової платформи.

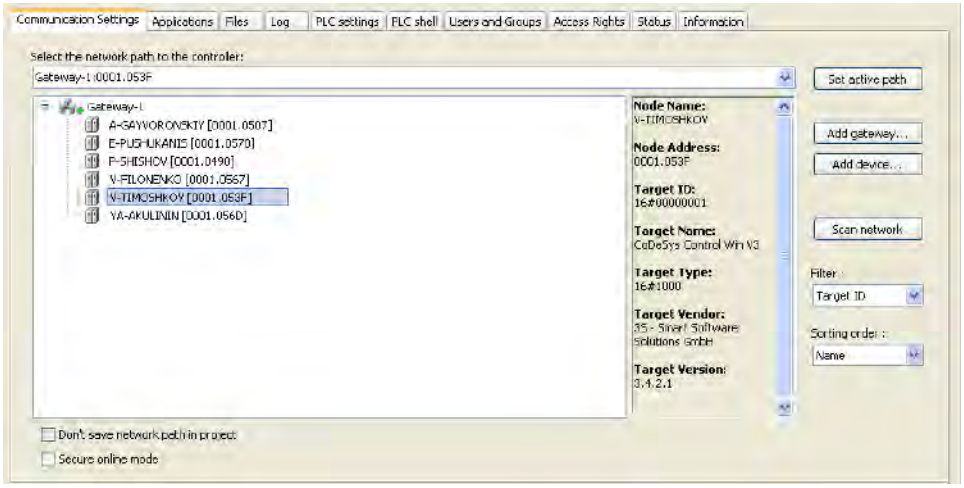


Рисунок 6.16 – Пошук віртуального контролера

Далі натисканням кнопки Set active path зробимо вибір використовуваного контролера. Праворуч від імені пристрою з'явиться напис active, як показано на рис. 6.17.

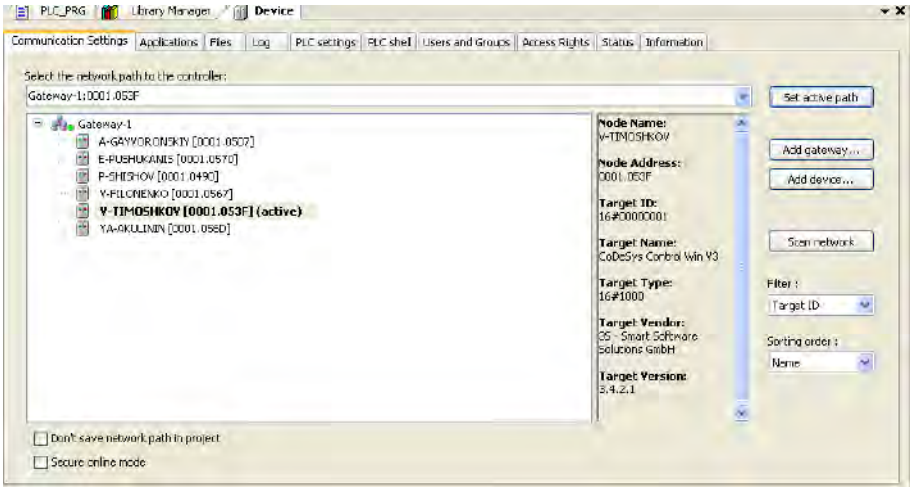


Рисунок 6.17 – Вибір активного контролера

Віртуальний контролер дозволяє не тільки робити налаштування програми, але й може бути використаний як цільова платформа для підключення зовнішніх пристроїв і взаємодії з ними за COM-портами ПК.

В наступних розділах віртуальній контролер використовуватиметься для доступу до віртуальних макетів за допомогою протоколу Modbus.

6.2 Реалізація обміну даними з пристроями за допомогою протоколу Modbus в середовище CODESYS

6.2.1 Створення та налаштування каналу Modbus

Використання протоколу Modbus в середовищі CODESYS потребує виконання наступних умов:

– додавання до проєкту нового пристрою, що реалізує доступ до мережі, наприклад Ethernet;

– створення майстра мережі та реалізація підлеглого пристрою;

– налаштування відповідних каналів для доступу до необхідних ресурсів підлеглого пристрою;

– налаштування віртуального ПК, як посередника між середовищем CODESYS, та віртуальним макетом, або пристроєм;

– налаштування підлеглого пристрою для сумісної роботи з CODESYS;

– виконання сумісного підключення до мережі віртуального макету та CODESYS.

Розглянемо зазначені пункти більш детально. На першому етапі створюємо новий проєкт, як описано в п.п. 6.1.1.

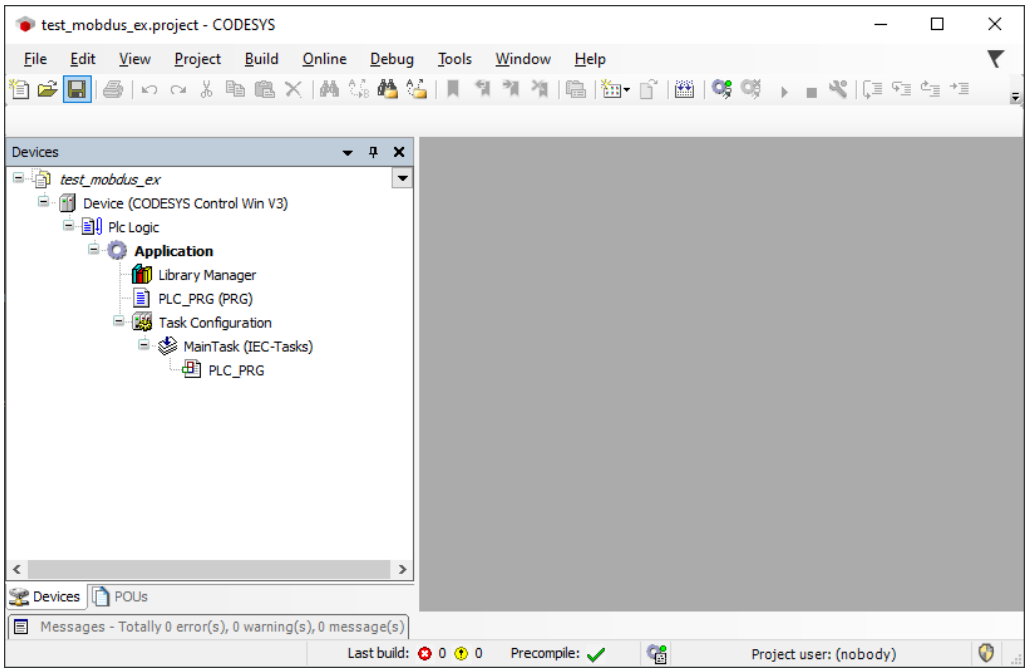


Рисунок 6.17 – Новий проєкт «test_modbus»

Після створення проєкту необхідно виконати додавання нового пристрою. Порядок операцій обирання нового пристрою подано на рис. 6.18.

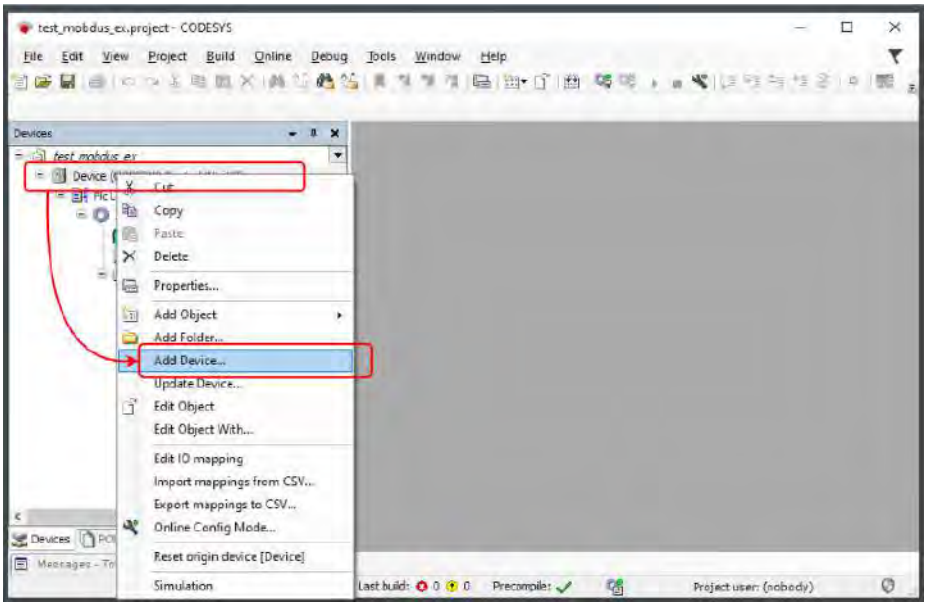


Рисунок 6.18 – Меню додавання нового пристрою

Для цього потрібно натиснути правою кнопкою миші на пункті «Device» в дереві проєкту та в меню, що відкриється, обрати «Додати новий пристрій».

У вікні, що відкриється (рис. 6.19) необхідно відкрити розділ «Ethernet-адаптер», а далі обрати пристрій «Ethernet».

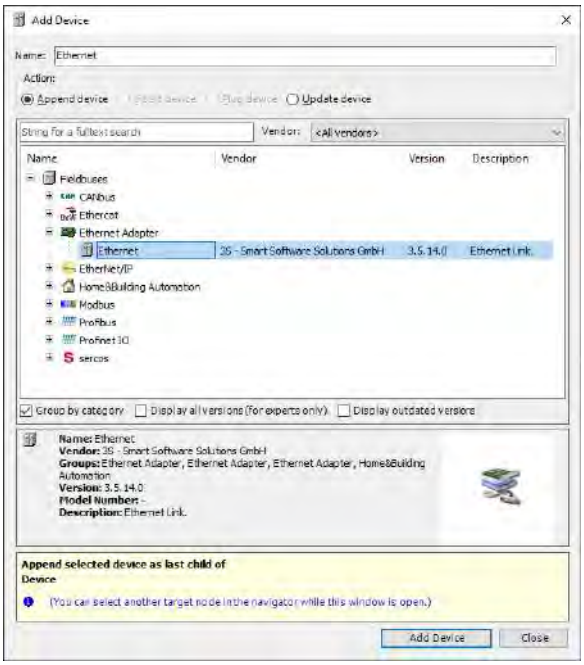


Рисунок 6.19 – Вибір пристрою «Ethernet»

Після натискання на кнопку «Додати пристрій» в дереві проєкту з’явиться новий пристрій (рис. 6.20).

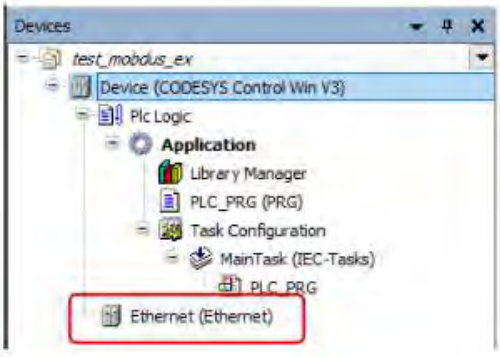


Рисунок 6.20 – Додавання нового пристрою до дерева проєкту

На наступному етапі створюємо майстра мережі. Для цього послідовно виконуємо чотири кроки (рис. 6.21):

– обираємо пристрій «Ethernet» в дереві проєкту (1);

– розкриваємо список Modbus-пристроїв (2);

– розкриваємо список майстер-пристроїв (3);

– обираємо пристрій «Modbus TCP Master» (4).

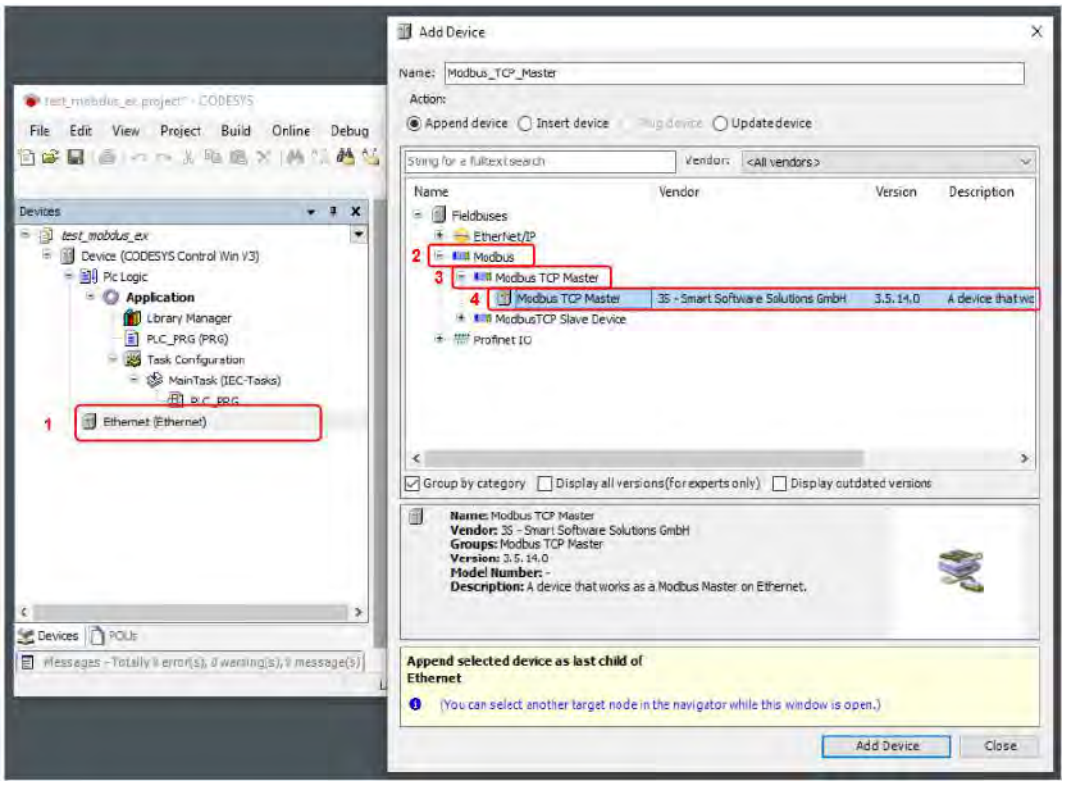


Рисунок 6.21 – Додавання майстра Modbus TCP в проєкт

Після натискання на кнопку «Додати пристрій» в дереві проєкту з’явиться майстер мережі (рис. 6.22).

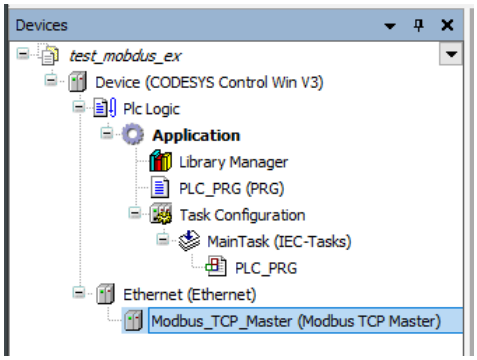


Рисунок 6.22 – Новий пристрій «Modbus_TCP_Master» в дереві проєкту

На наступному етапі в дереві проєкту необхідно обрати пункт «Modbus_TCP_Master» (1) (рис. 6.23), відкрити вікно додавання нового пристрою, та послідовно обрати:

– пункт «Modbus» (2);

– Slave Modbus TCP (3);

– Modbus TCP Slave (4).

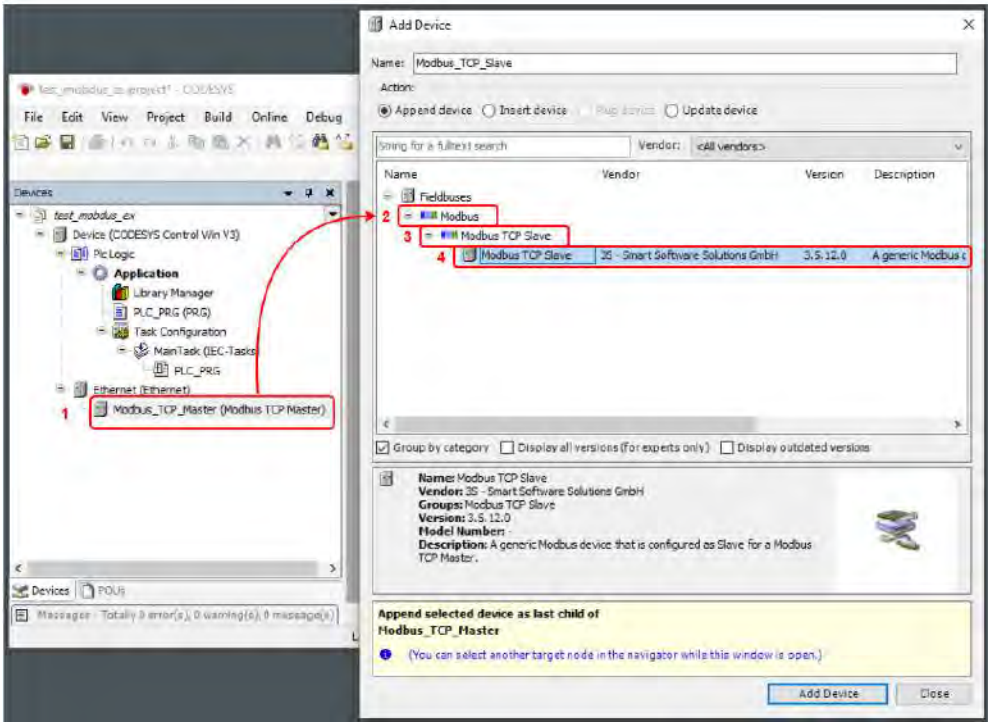


Рисунок 6.23 – Додавання Modbus TCP Slave Device

Таким чином, після натискання на кнопку «Додати пристрій» до проєкту буде додано підлеглий пристрій, що асоціюватиметься з пристроєм, яким керуватимемо в проєкті. Дерево проєкту повинно виглядати, як подано на рис. 6.24. Обов'язково потрібно перевірити, щоб підлеглий пристрій входив до складу майстра.

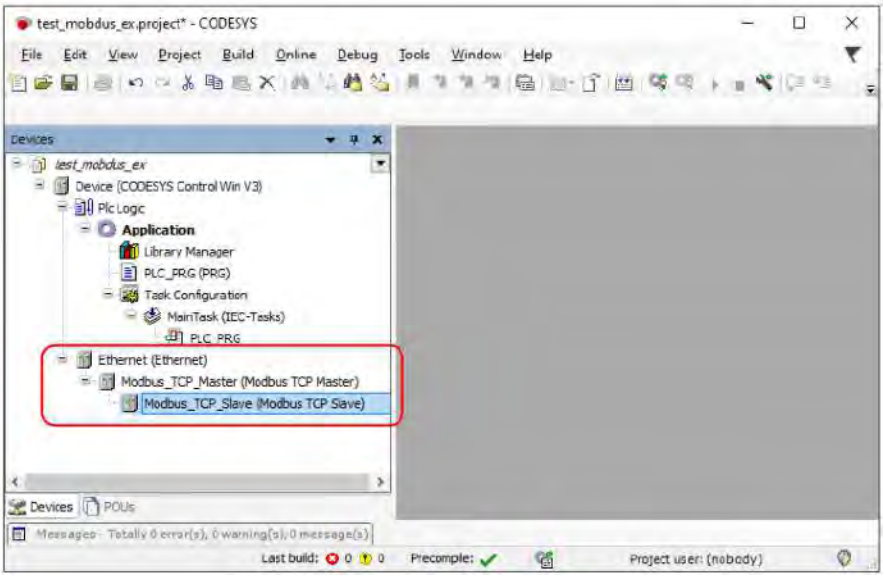


Рисунок 6.24 – Дерево проєкту після додавання майстра та підлеглого Modbus

Якщо два цих пристрої знаходяться на одному рівні в ієрархії дерева проєкту, то це буде помилкою і така структура працювати не буде. Приклад помилкової структури подано на рис. 6.25.

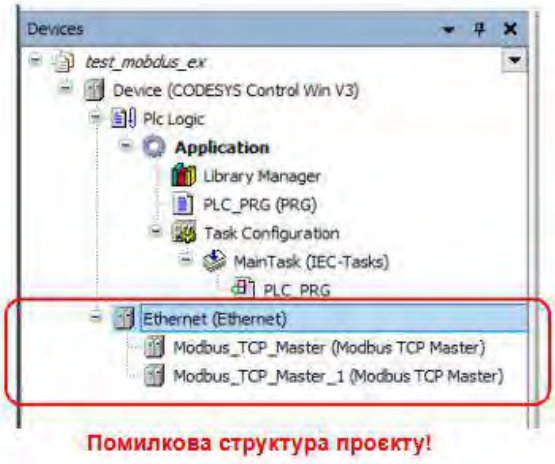


Рисунок 6.25 – Помилкова структура проєкту

Після додавання нових пристроїв необхідно виконати налаштування з’єднання з підлеглим пристроєм. Це робиться у вікні загальних параметрів протоколу Modbus середовища Codesys (рис. 6.26).

В даній вкладці вводиться IP-адреса підлеглого пристрою. IP-Адресу можна дізнатись в параметрах налаштування мережі Ethernet комп’ютера, де завантажується, наприклад, віртуальний макет світлової колони. В даному випадку ця адреса: 192.168.1.2.

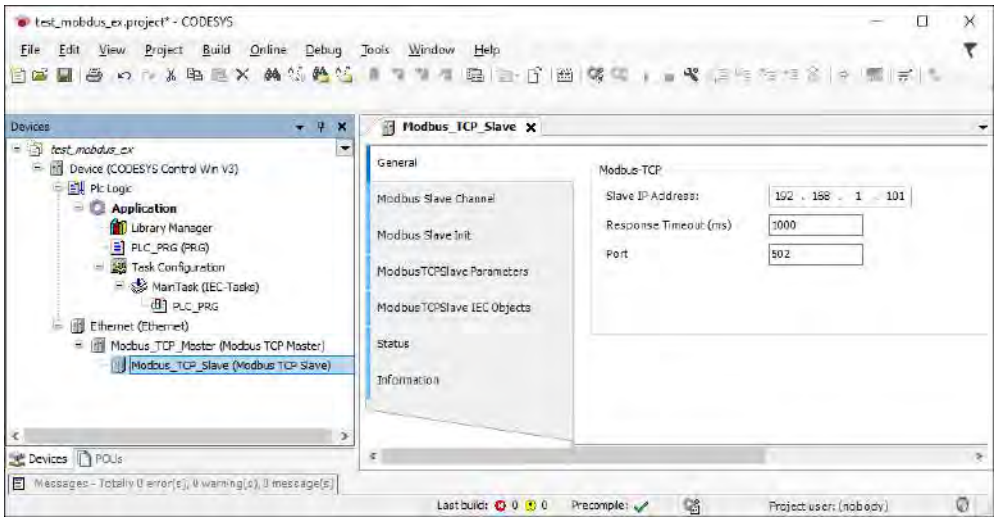


Рисунок 6.26 – Загальні параметри протоколу Modbus середовища CODESYS

Також потрібно перевірити номер порту для доступу до мережі Modbus. В даному прикладі – це 502 порт.

Ще одним параметром є час очікування відповіді від підлеглого пристрою, після якого генерується помилка з’єднання. За замовчуванням це значення дорівнює 1000 мс.

Далі виконується створення та налаштування відповідних каналів для доступу до необхідних ресурсів підлеглого пристрою. Такі налаштування необхідно проводити для кожного ресурсу.

Наприклад, якщо необхідно керувати світловою колоною віртуального макету «Traffic Light», спершу визначають за яким принципом відбуватиметься управління світлодіодними сегментами. В разі, якщо до кожного сегменту передбачається окремий доступ, то необхідно буде створити три незалежних канали Modbus (функція 05). У випадку, якщо управління сегментами буде відбуватись групою (функція 0F), потрібно створити лише один канал Modbus.

В останньому випадку кожен раз потрібно буде передавати інформацію відразу для всіх сегментів, навіть, якщо потрібно змінити стан лише одного.

Для прикладу створимо канал для доступу до одного жовтого сегменту світлової колони з адресою 00002 (рис.6.27).

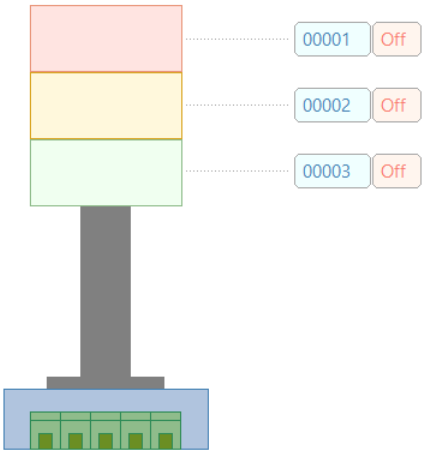


Рисунок 6.27 – Зображення світлової колони та адреси її сегментів

В CODESYS адреса ресурсу в мережі Modbus задається у вигляді зміщення від нульової адреси. Адреса червоного сегменту вважається за нульову (00001), тому адреса другого, жовтого сегменту записуватиметься, як зміщення +0x0001 відносного початку адресного простору вихідних контактів.

Виклик діалогового вікна створення каналу Modbus відбувається натисканням кнопки «Додати канал» у вкладці «Канал Modbus Slave» (рис. 6.28).

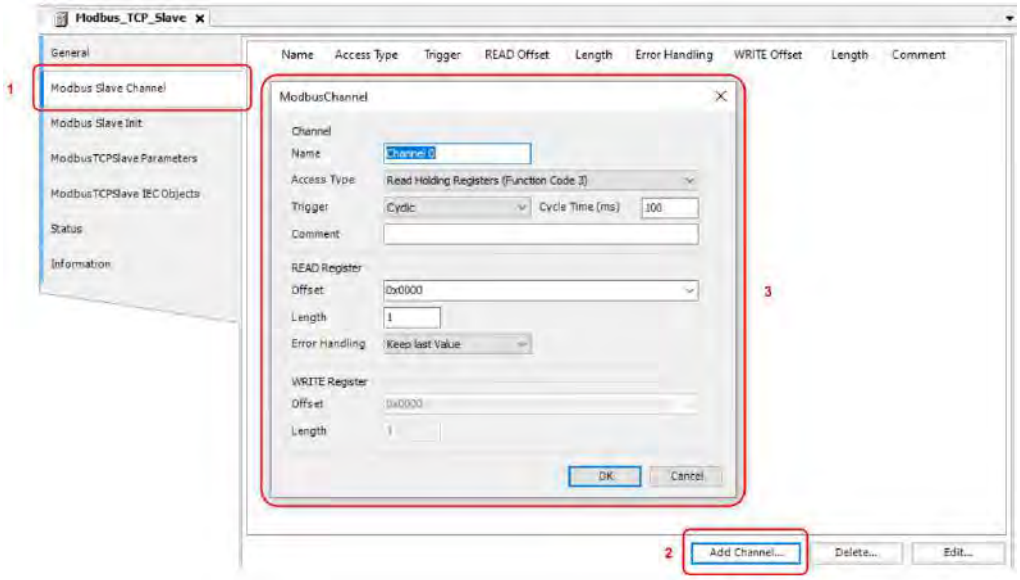


Рисунок 6.28 – Створення нового каналу Modbus

У вікні опису каналу Modbus необхідно:

– ввести назву каналу;

– обрати тип функції доступу до ресурсів підлеглого пристрою;

– ввести коментар для опису створеного каналу та полегшення його ідентифікації в наборі інших каналів;

– вказати зміщення адреси ресурсу підлеглого пристрою відносно початкового значення.

На рис. 6.29 подано приклад заповнення характеристик каналу Modbus для керування жовтим сегментом світлової колони.

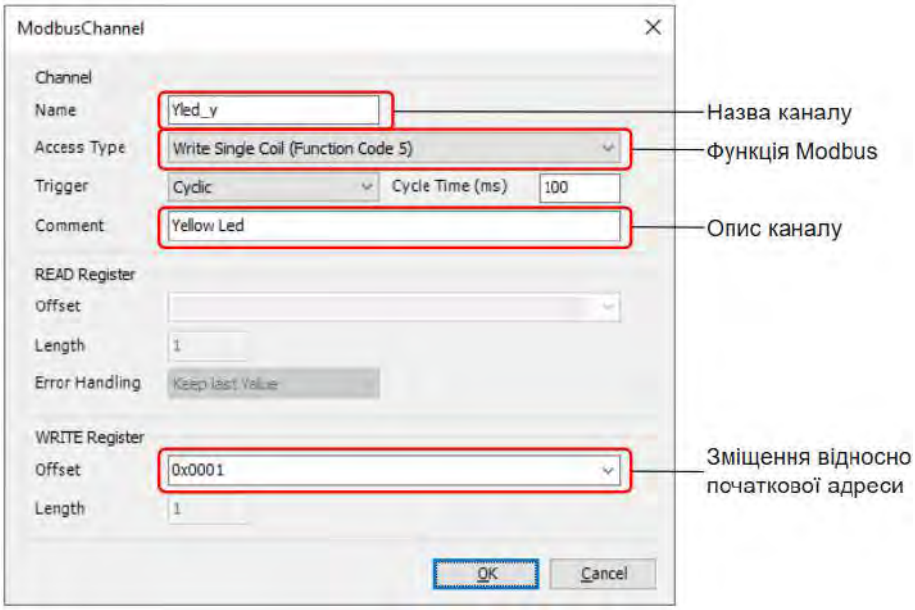


Рисунок 6.29 – Приклад заповнення характеристик каналу Modbus для керування жовтим сегментом світлової колони

В якості назви каналу обрано «Yled_y». Ця назва співпадає з назвою змінної в програмі віртуального макета. Для доступу до єдиного сегмента колони обрано тип функції (05 – запис одного вихідного контакту).

В полі коментаря вказано «Yellow Led» для пояснення, для якого саме ресурсу створено цей канал.

Як було зазначено вище, жовтий сегмент розташовується другим у списку ресурсів підлеглого пристрою, тому в якості зміщення відносно початкової адреси вказано число «0x0001».

Для роботи зі створеним каналом із програми можна посилатись на нього через створені змінні (рис. 6.30).

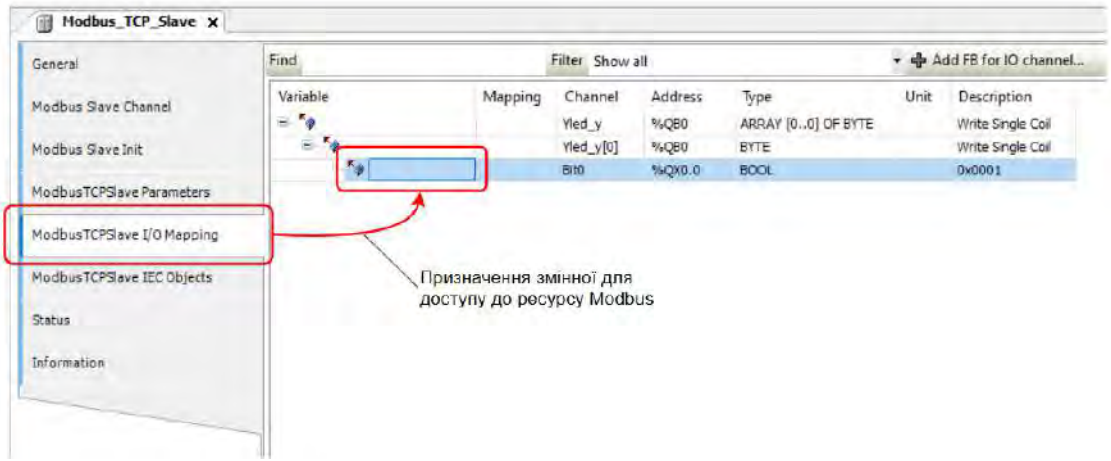


Рисунок 6.30 – Призначення змінної для доступу до ресурсів Modbus

Для призначення змінної існує декілька шляхів:

– створення змінної в проєкті та вибір її у вікні конфігурації (рис. 6.31);

– безпосереднє введення назви змінної в полі відповідності ресурсам Modbus (рис. 6.32).

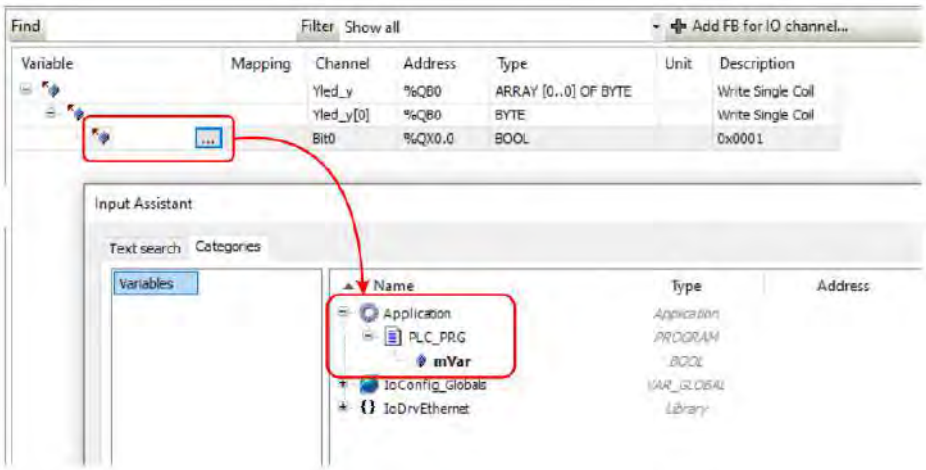


Рисунок 6.31 – Вибір змінної з наявних в проєкті

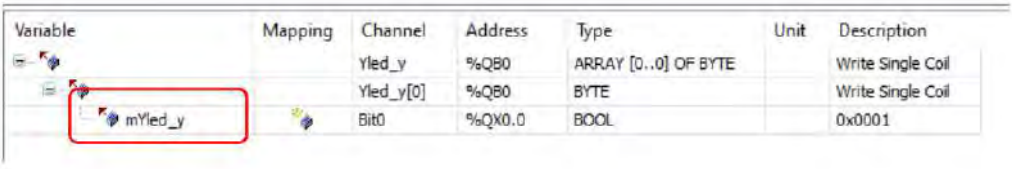


Рисунок 6.32 – Створення нової змінної

В другому варіанті нова змінна потрапить в розділ глобальних змінних проєкту в неявному вигляді.

Докладно про правила створення та прив’язки змінних в проєкті CODESYS до ресурсів вводу / виводу подано в пункті 6.3 даного посібника.

На даному етапі налаштування каналів Modbus завершено, та можна перейти до тестування з’єднання з віртуальним приладом.

6.2.2 Налаштування з’єднання з віртуальними приладами

Доступ до віртуальних приладів відбувається за допомогою шлюзу CODESYS Gateway та віртуального ПЛК (див. п. 6.1.3).

Для поєднання проєкту CODESYS з віртуальним приладом необхідно виконати наступний порядок дій.

Запустити віртуальний прилад, обрати потрібний макет та виконати налаштування підключення до мережі Modbus за протоколом TCP/IP (рис. 6.33).

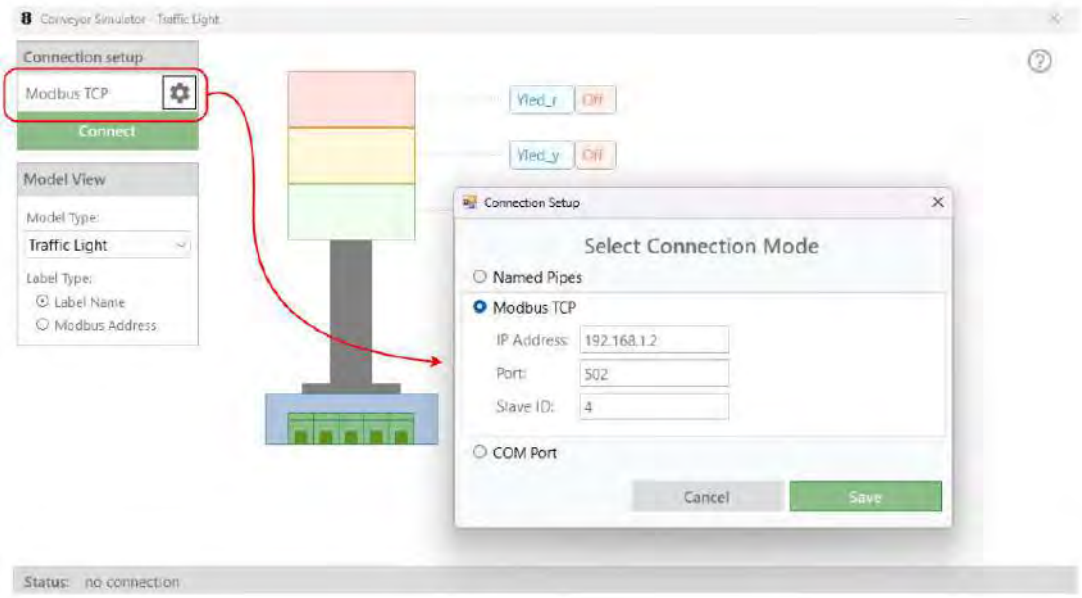


Рисунок 6.33 – Конфігурація підключення до мережі Modbus

Після завершення конфігурації потрібно натиснути кнопку «Connect» та перевести віртуальний пристрій в режим очікування команд від майстра мережі.

Для перевірки принципів взаємодії двох пристроїв за протоколом Modbus напишемо просту демонстраційну програму, що буде вмикатиме та вимикатиме жовтий сегмент світлової колони через кожну секунду.

Частина програми з описом змінних може мати такий вигляд:

```
PROGRAM PLC_PRG
VAR
    mBool_4:BOOL;
    mWord_1:WORD;
    Delay: TON;
END_VAR
```

Змінна Delay типу TON використовується для організації затримки часу в програмі. Вона має три важливі параметри:

– IN: ознака ввімкнення, або вимкнення даного блоку;

– PT: параметр, що задає час затримки;

– Q: ознака, що вказує на те, що зазначений час сплинув.

Створимо програму, яка використовуватиме зазначений вище елемент затримки та змінювати стан обраного сегменту світлової колони для подання візуальних сигналів:

```
Delay(IN:=TRUE, PT:=T#1S);
IF NOT(Delay.Q) THEN
    RETURN;
END_IF
Delay(IN:=FALSE);

Yled_y:= NOT Yled_y;
```

Рядок Yled_y:= NOT Yled_y змінює стан змінної «Yled_y» на протилежний, кожного разу, коли програма його виконує.

Для управління віртуальним макетом необхідно спочатку запустити віртуальний ПЛК (CODESYS SP Win V3). Запускається він натисканням правої кнопки миші на іконці, яка розміщена на панелі задач Windows (рис. 6.34).

Віртуальний ПЛК надасть доступ до ресурсів операційної системи та до мережі Ethernet, в якій запущено віртуальний макет світлової колони.

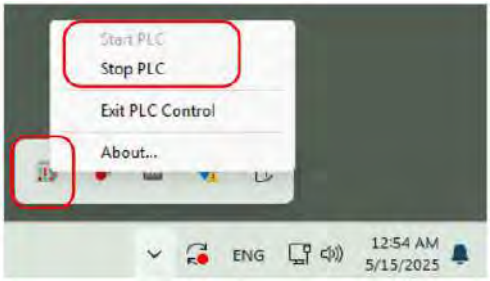


Рисунок 6.34 – Запуск віртуального ПЛК

Далі, двійним кліком на опції «Device» в проєкті CODESYS, викликаємо вікно налаштування підключення до віртуальних пристроїв (рис. 6.35).

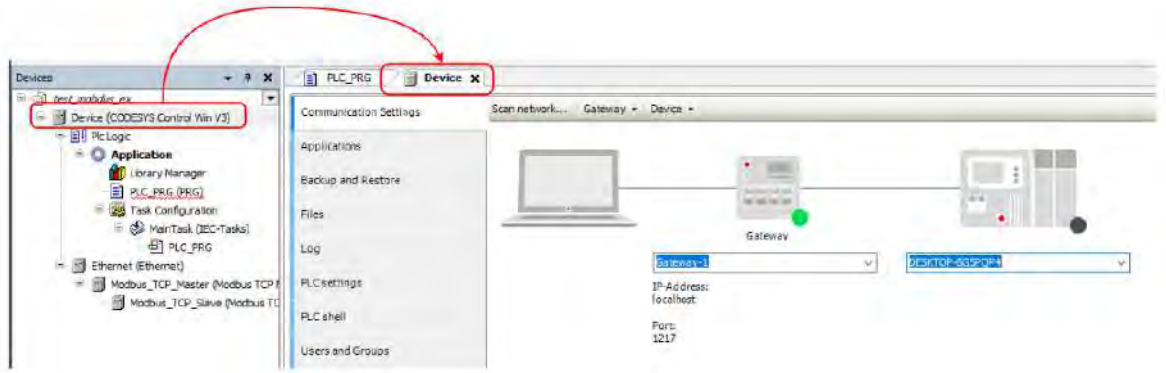


Рисунок 6.35 – Вікно підключення до віртуальних пристроїв

Відкриваємо вікно «Сканувати мережу» (рис. 6.36) та обираємо доступний шлюз зі списку наявних, або виконуємо новий пошук.

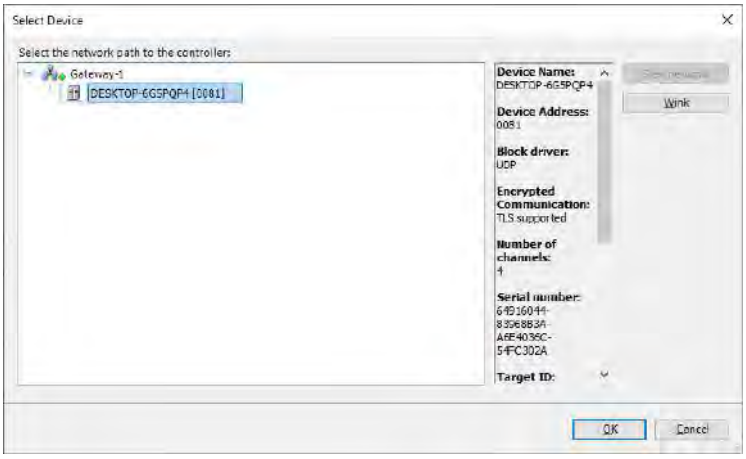


Рисунок 6.36 – Обираємо доступний шлюз зі списку наявних

Після натискання на кнопку «ОК» виконується підключення до обраного пристрою (рис. 6.37).

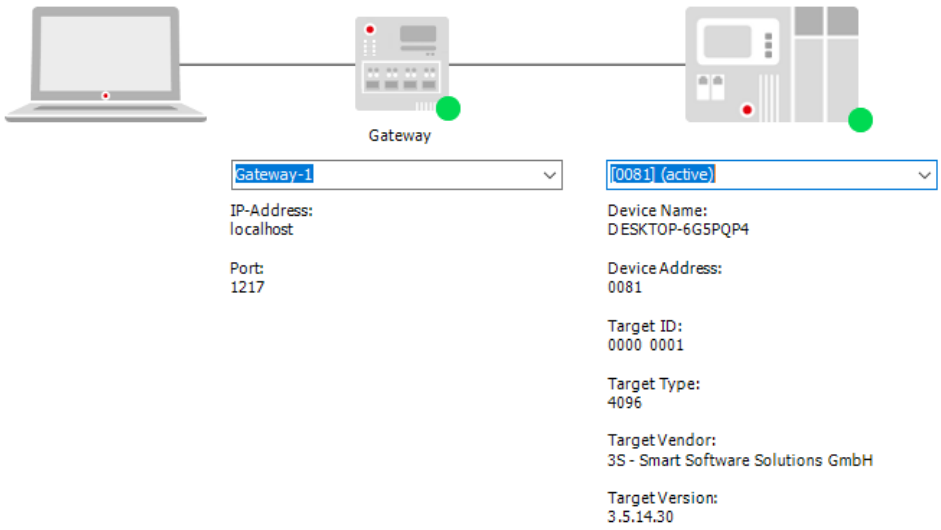


Рисунок 6.37 – Успішне підключення до обраного пристрою

Ознакою вдалого підключення – є зелена мітка під зображенням пристрою в правій частини вікна.

Виконуємо компіляцію програми, завантаження коду в віртуальний ПЛК та запуск програми на виконання. На рис. 6.38 подано результат сумісної роботи CODESYS та віртуального макету.

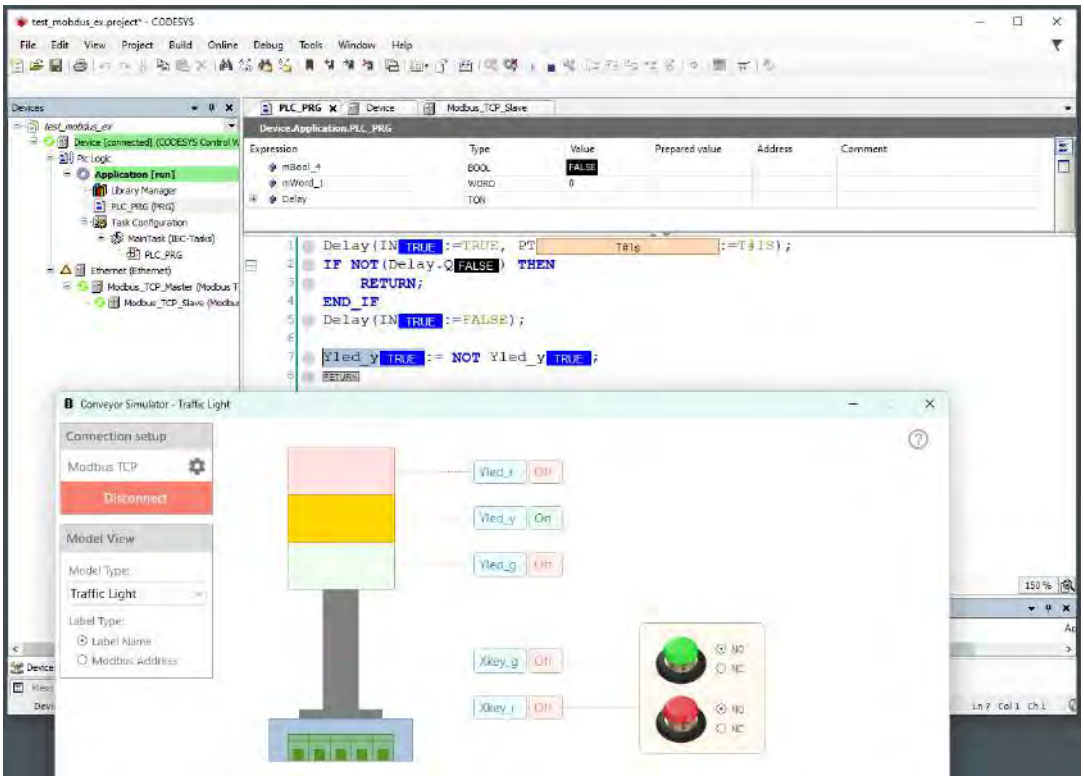


Рисунок 6.38 – Результат сумісної роботи CODESYS та віртуального макету

Ознакою правильної роботи мережі Modbus є дві зелених позначки навпроти пристроїв Modbus_TCP_Master, та Modbus_TCP_Slave (рис. 6.39).

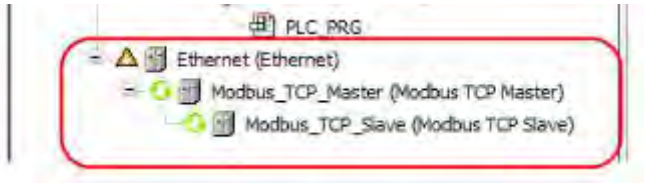


Рисунок 6.39 – Ознака правильної роботи мережі Modbus

В разі нештатної поведінки віртуального приладу можна перезавантажити Software PLC, виконавши послідовно його вимкнення, та, знову, увімкнення.

6.3 Правила створення та прив'язки змінних в проєкті CODESYS до ресурсів вводу / виводу

6.3.1 Налаштування пристроїв

В середовищі CODESYS існує можливість налаштувати об'єкти пристрою, що додані в дерево пристроїв, у відповідному редакторі. Можливості налаштування залежать від опису пристрою в конфігураторі. «Загальний редактор пристроїв» надає вкладки, які за потреби доповнюються вкладками для певного пристрою.

Для доступу до налаштування портів вводу / виводу необхідно відкрити проєкт, у дерево пристроїв якого додано ПЛК, або інші пристрої, розкрити цей об'єкт, та під ним знайти потрібні ресурси пристрою польової шини (рис. 6.40).

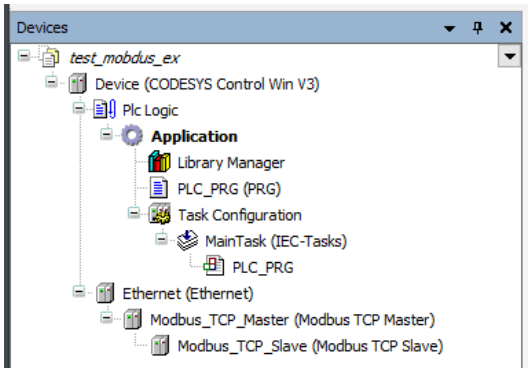


Рисунок 6.40 – Дерево ресурсів пристрою

Далі необхідно двічі клацнути на об’єкті пристрою у дереві для відкриття відповідної вкладки (рис. 6.41).

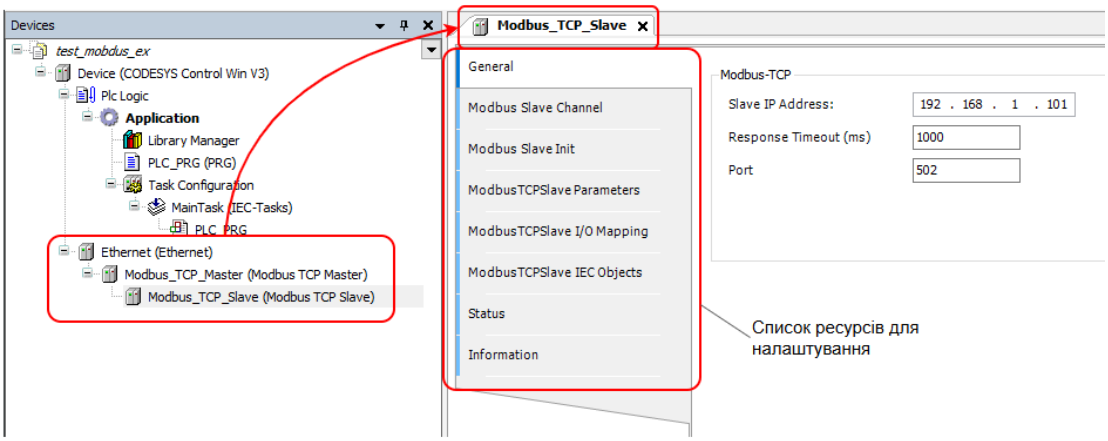


Рисунок 6.41 – Вибір ресурсу для налаштування

Для кожного пристрою перелік доступних ресурсів буде іншій. Серед доступних опцій потрібно обрати пункт налаштування портів вводу / виводу та відкрити його для прив’язки змінних до ресурсів пристрою.

6.3.2 Загальна інформація про відображення вводу / виводу

Можливість налаштування «відображення» вводу / виводу для змінних проєкту або навіть для цілих функціональних блоків, залежить від типу пристрою. Налаштування карти вводу / виводу означає зв'язування каналів входу та виходу пристрою зі змінними проєкту. Для цього також використовується термін «відображення».

Існує декілька правил роботи з відображенням входів і виходів пристрою на змінні в CODESYS:

– програма не має доступу на запис до змінних, які відображаються як вхід;

– зіставляти існуючу змінну можна лише з одним входом або виходом;

– можна безпосередньо генерувати нові глобальні неявні змінні в карті вводу-виводу та відображати їх у каналі пристрою;

– схема пам'яті структур визначається пристроєм;

– можна змінювати адреси та зберігати ці значення в карті вводу / виводу;

– зміни в карті вводу / виводу можуть бути передані на контролер під час завантаження програми.

Для кожної змінної, яка призначена каналу вводу / виводу в діалоговому вікні відображення вводу / виводу, можна створити «примусові змінні» в процесі компіляції програми. Використовуючи ці змінні, можна, наприклад, під час введення в експлуатацію установки, примусово встановити значення на вході або виході через візуалізацію / HMI.

Якщо використовується покажчик на вхід пристрою, доступ вважається доступом для запису, наприклад

pTest := ADR(input);.

Це призводить до попередження компілятора під час генерації коду: "...invalid assignment target". Якщо потрібна конструкція такого типу, спочатку необхідно скопіювати вхідне значення в змінну з доступом на запис.

Адреса вводу / виводу також може бути пов'язана зі змінною за допомогою «декларації AT» в форматі IEC. Однак, оскільки конфігурація пристрою часто змінюється, рекомендується виконувати призначення лише в редакторі пристроїв.

Якщо використовується декларація AT, необхідно звернути увагу на наступне:

– декларація AT допустима лише для локальних або глобальних змінних, а не для вхідних або вихідних змінних функціональних блоків;

– неявні «примусові змінні» для вводу / виводу не можуть бути згенеровані для декларацій AT;

– якщо використовується декларація AT зі структурними змінними або змінними функціонального блоку, усі екземпляри матимуть доступ до однієї області пам'яті. Тоді це відповідає використанню «статичних змінних» у класичних мовах програмування, таких як «C».

6.3.3 Особливості прямої адресації

Для створення змінної, що прямо адресується, використовується таке оголошення:

ім'я змінної AT %пряма адреса тип;

Пряма адреса починається з літери, яка визначає область пам'яті, як подано в таблиці 6.1.

Таблиця 6.1 – Визначення області пам'яті

Символ	Область пам'яті
I	Область входів
Q	Область виходів
M	Пам'ять, що прямо адресується

Далі йде символ, який визначає тип прямої адреси. У таблиці 6.2 подано повний список символів, що визначає тип прямої адреси.

Таблиця 6.2 – Повний список символів, що визначає тип прямої адреси

Символ	Область пам'яті
ні	Біт
X	Біт
B	Байт
W	Слово
D	Подвійне слово
L	Довге слово

Завершує пряму адресу число – складена ієрархічна адреса, поля якої розділені крапкою. У найпростішому випадку використовується два поля адреси: номер елемента і номер біта. На рис. 6.42 подана схема побудови імені змінної, що прямо адресується.

В кінці оголошення, як і для змінних, що автоматично розміщуються, необхідно вказати тип змінної. Із зазначенням адреси одного біта тип змінної може бути тільки BOOL.

У прямій адресі вказується саме номер елемента. Це докорінно відрізняється від фізичних адрес мікропроцесора. Якщо пряму адресу визначає байт, то номер елемента – це номер байта. Якщо пряму адресу визначає слово, то номер елемента – це номер слова, і, відповідно, один елемент займає два байти.

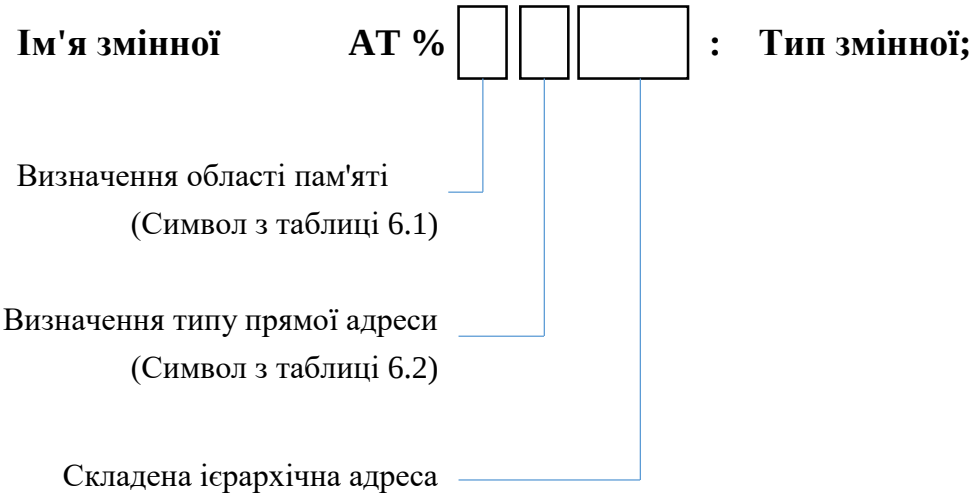


Рисунок 6.42 – Схема побудови імені змінної, що прямо адресується

Так, наступні три записи адресують один і той самий байт:

```
dwHeat    AT %MD1:    BYTE;
wbHeat    AT %MW2:    BYTE;
byHeat    AT %MB4:    BYTE;
```

Нумерацію елементів пам'яті для даного прикладу ілюструє таблиця 6.4.

Таблиця 6.4 – Приклад розподілу змінних у пам'яті

D	0				**1**			
W	0		1		**2**		3	
B	0	1	2	3	**4**	5	6	7

На випадок, якщо така схема виявиться чомусь неприйнятною, компілятор CODESYS має спеціальний прапорець, який примусово вмикає байтову адресацію для всіх типів.

У кожній області пам'яті адресація елементів починається з нуля. Фізичне розміщення областей, які прямо адресуються в ОЗП, визначається конфігурацією контролера.

Очевидно, що зіставлення ідентифікаторів змінних прямим адресам є справою, що вимагає акуратності. Тому для складних модульних контролерів застосовуються спеціальні фірмові конфігуратори, що підключаються до оболонки комплексу програмування і дозволяють графічно «зібрати» ПЛК і визначити всі необхідні інтерфейсні змінні.

Входи ПЛК – це змінні з прямими адресами в області I. Вони доступні в прикладних програмах тільки за читанням. Виходи Q – тільки за записом. Змінні в області M доступні за записом і читанням.

В області пам'яті M розміщують зазвичай змінні, які не можна однозначно віднести до входів або виходів. Це можуть бути діагностичні ресурси модулів, параметри системи виконання тощо.

Прямі адреси можна використовувати в програмах безпосередньо, наприклад:

```
IF %IW4 > 1 THEN ...    (*Значення входу IW4*)
```

Проте, все ж бажано компактно зосередити в проєкті всі апаратно-залежні моменти.

Зверніть увагу, що пряма адресація дозволяє розмістити кілька різнотипних змінних в одній пам'яті. Наприклад, спеціально для швидкого обнулення 16-дискретних виходів (BOOL), можна використовувати змінну типу WORD. Або, наприклад, поєднати змінну STRING і декілька змінних типу BYTE, що дасть можливість організувати форматування виводу без застосування рядкових функцій. Оскільки фізичний розподіл адрес відомий на етапі трансляції, компілятор формує максимально компактний код для таких об'єднань, чого не вдається досягти у роботі з елементами масиву, де потрібна динамічна адресація.

6.3.4 Пов’язування ресурсів пристрою з існуючою змінною проєкту

Пристрій, який підтримує конфігурацію відображення вводу / виводу в CODESYS, вставляється в дерево пристроїв проєкту, таким чином, на вкладці I/O Mapping у редакторі пристроїв буде представлено табличне відображення вхідних і вихідних каналів пристрою з вказанням адрес і типів даних.

При зіставленні «занадто великих» типів даних, якщо розмір типу змінної перевищує байт, то змінна зіставляється з байтовою адресою, а її значення буде

скорочене до розміру байту! В процесі моніторингу значення змінної в діалоговому вікні відображення вводу/виводу у кореневому елементі адреси відображається значення, яке наразі має змінна в проєкті. Поточні окремі бітові значення байту послідовно відображаються в бітових елементах під ним, але цього може бути недостатньо для розуміння всього значення змінної.

Виконання даного типу прив'язки потребує виконання наступних кроків.

Наприклад, в POU оголосимо, дві змінні: mBool_4 типу BOOL, та mWord_1 типу WORD. За допомогою цих змінних необхідно отримати доступ до вихідних даних цільового пристрою з програми (рис. 6.43).

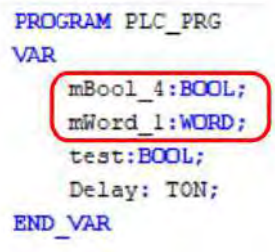


Рисунок 6.43 – Створення змінної mBool_4 типу BOOL

Далі необхідно двічі клацнути на об'єкт пристрою в дереві пристроїв, щоб відкрити редактор, а потім вкладку «Відображення вводу / виводу» з ім'ям пристрою.

У таблиці ресурсів пристрою необхідно знайти стовпець «Змінна» з відображенням каналів вводу та виводу пристрою. Припустимо, що є пристрій вводу типу WORD. Він відображається з окремими бітовими адресами (бітовими каналами) під вузлом WORD (рис. 6.44).

Під час відображення структурних змінних, редактор не дозволяє вводити як структурну змінну (приклад: %QB0), так і окремі елементи структури (приклад: %QB0.1 і QB0.2). Таким чином, якщо в таблиці відображення є головний запис із піддеревом записів бітових каналів, тоді застосовується наступне: можна вказати змінну або в рядку головного запису, або в рядках піделементів (бітові канали), але не в обидва.

Тепер можна прив'язати змінні до ресурсів зайнявши або весь канал змінною відповідного типу, АБО його окремі бітові адреси каналу відповідними змінними типу BOOL або BIT.

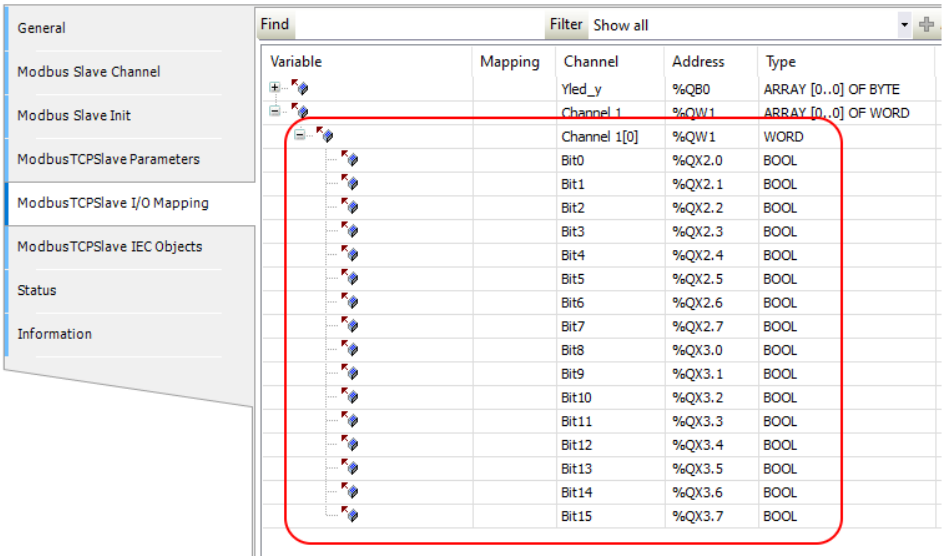
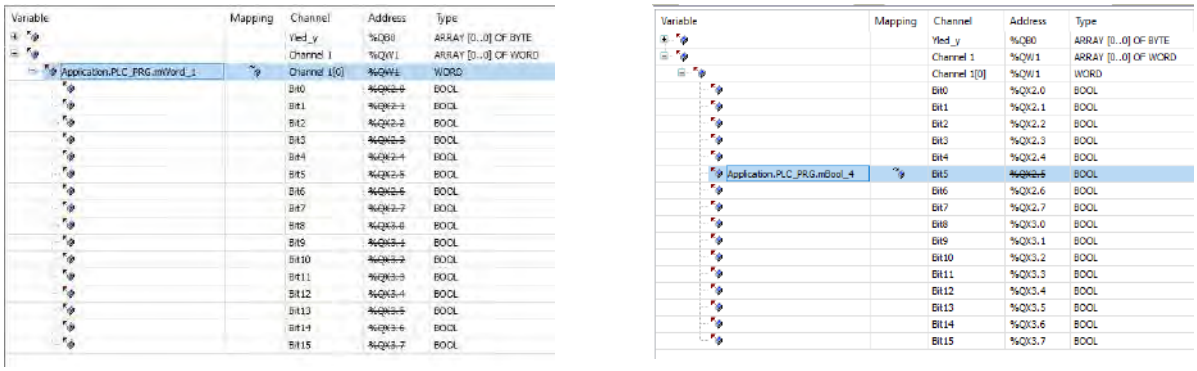


Рисунок 6.44 – Відображення ресурсів пристрою

Наприклад, змінна mWord_1 типу WORD може бути пов’язана відразу зі всім каналом Cannel1, тому що її тип передбачає використання 16 біт даних (рис. 6.45, а). Або можемо використати тільки один біт ресурсу, прив’язавши змінну mBool_4 типу BOOL до будь-якого бітового каналу (рис 6.45, б).



а) б)

Рисунок 6.45– Прив’язка змінної до ресурсів пристрою

Для обирання змінної та прив’язки її до відповідного ресурсу необхідно два рази клікнути на порожньому місці в рядку потрібного ресурсу в стовбчику «Змінна» та натиснути на кнопку з трьома крапками. що з’явиться збоку (рис. 6.46).

Рисунок 6.46 – Кнопка виклику вікна обирання змінної

У вікні, що відкриється потрібно розкрити пункт «Application», обрати необхідний програмний модуль, наприклад, PLC_PRG, та зі списку наявних змінних обрати потрібну (рис. 6.47).

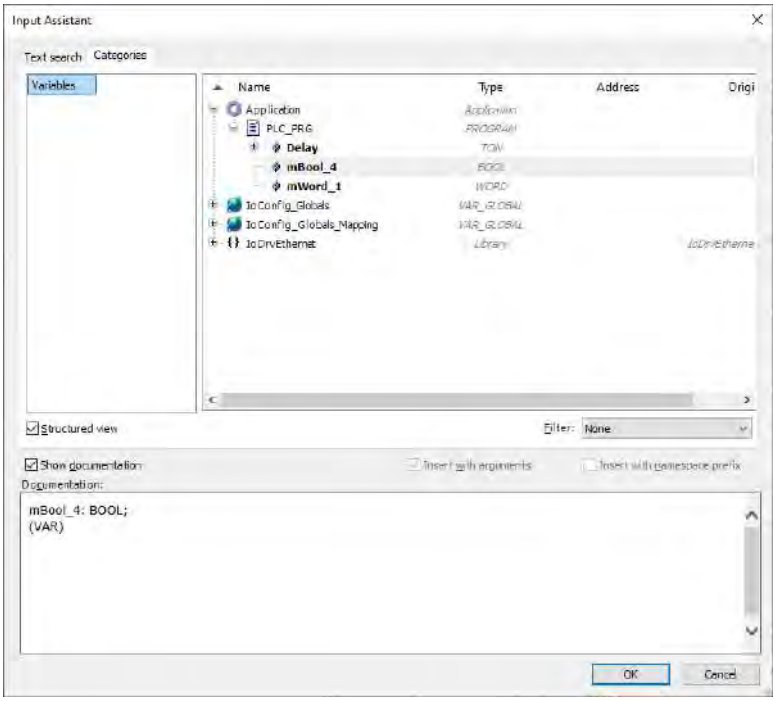


Рисунок 6.47 – Обирання змінної зі списку наявних

Ознакою закріплення змінної є символ HMI ( ), що відображається в стовпці «Відображення». Закреслена адреса показує, що цю адресу більше не можна застосовувати в інших частинах проєкту, щоб уникнути невизначеності проведінки програми.

Для версії компілятора V3.5 SP11 і пізнішої версії значення ініціалізації змінних використовується автоматично як значення за замовчуванням під час зіставлення з існуючою змінною. Є можливість редагувати поле «Значення за замовчуванням», лише якщо виконується зіставлення нової створеної змінної або, якщо зіставлення не вказано. У старіших версіях користувачі повинні були явно вказати, що значення за замовчуванням і значення ініціалізації ідентичні.

6.3.5 Пов'язування ресурсів пристрою з новою змінною проєкту

Якщо змінна створюється прямо на етапі виконання конфігурації ресурсів пристрою, то така змінна стає глобальною та неявною. Її явного відображення в проєкті в розділі опису змінних не буде.

Приклад створення такої змінної подано на рис. 6.48.

В процесі створення такої змінної відмінністю є те, що діалогове вікно обирання змінної не викликається, а її назва вводиться безпосередньо у полі введення навпроти відповідного ресурсу. Також відсутнім є елемент закреслення пов'язаного з нею ресурсом (рис. 6.48).

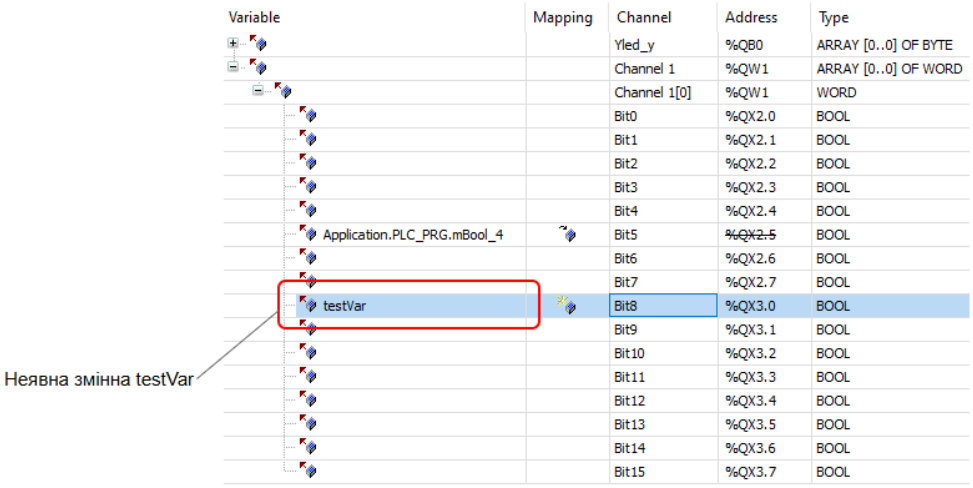


Рисунок 6.48 – Приклад створеної неявної змінної

Після створення такої змінної її можна знайти в списку глобальних у вікні асистента введення (рис. 6.49).

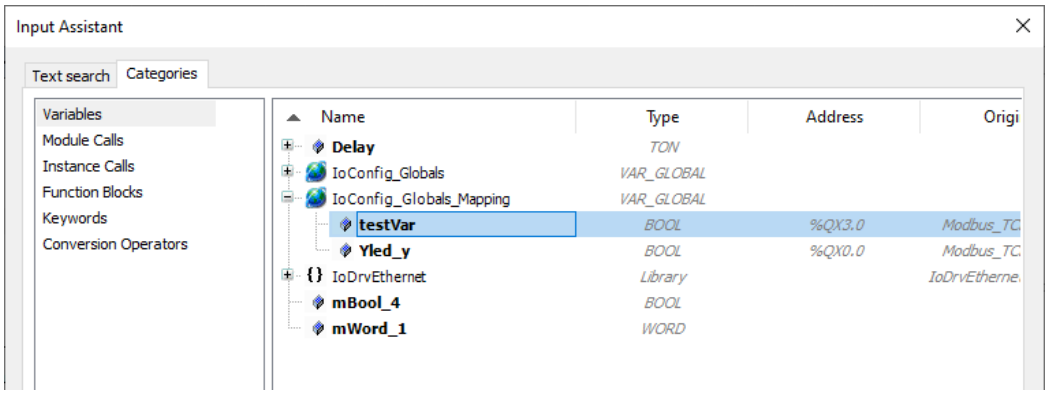


Рисунок 6.49 – Глобальна змінна у вікні асистента введення

6.3.6 Зміна та фіксація значення адреси в карті ресурсів вводу / виводу

CODESYS дозволяє змінювати значення адреси всього каналу (але не значення окремого піделемента каналу) у таблиці відображення на вкладці «Відображення вводу / виводу» певного пристрою. Це дозволяє адаптувати адресацію до заданої конфігурації машини та зберегти значення адреси, навіть якщо компонування модулів змінюється. За замовчуванням зміна макета призводить до автоматичної адаптації значень адреси.

Порядок дій наступний:

– обираємо пристрій для налаштування та переходимо до карти його ресурсів;

– обираємо потрібний ресурс та в полі «Адреса» переходимо до редагування адреси обраного ресурсу.

Після завершення редагування біля адреси ресурсу з’явиться ознака «M», що означитиме, що адреса модифікована (рис. 6.50).

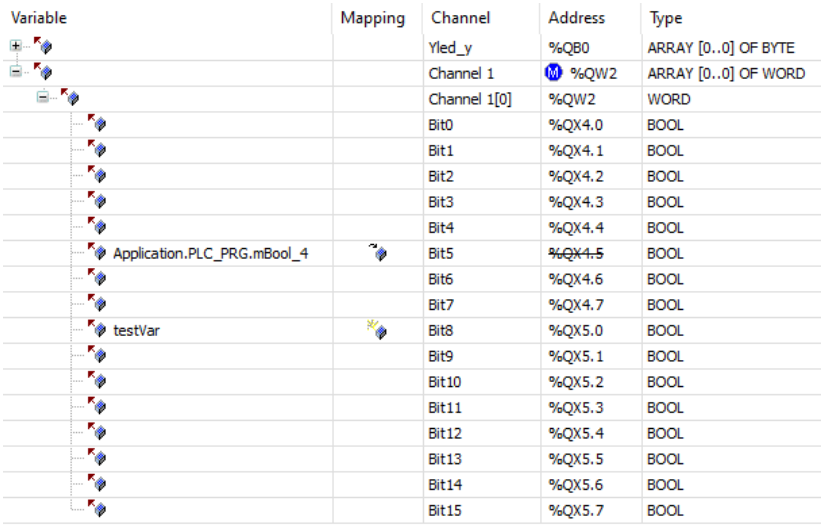


Рисунок 6.50 – Модифікація адреси ресурсу

Символ «M», що відображається перед адресою означає, що адреса фіксована. Відповідно змінюються і адреси піделементів каналу. Якщо тепер буде змінене положення об’єкта пристрою всередині інших об’єктів в дереві пристроїв, CODESYS автоматично не адаптує ці адреси до нового порядку, як це було б без фіксації адреси.

6.3.7 Генерація неявних змінних для примусового закріплення ресурсів вводу / виводу

Під час введення в експлуатацію установки або машини може виникнути необхідність «примусово» змінити значення на входах, або виходах. Якщо пристрій підтримує це, можна створити спеціальні «примусові змінні» для цієї задачі та використовувати їх, наприклад, у візуалізації HMI.

Для налаштування відкриваємо редактор пристроїв, вкладку налаштувань ПЛК, двічі клацнувши об’єкт пристрою в дереві пристроїв.

Активуємо параметр «Генерувати примусові змінні для відображення вводу / виводу». Цей параметр доступний лише, коли пристрій підтримує цю функцію. Наприклад, на рис. 6.51 подано приклад, коли пристрій не підтримує налаштування таких змінних.

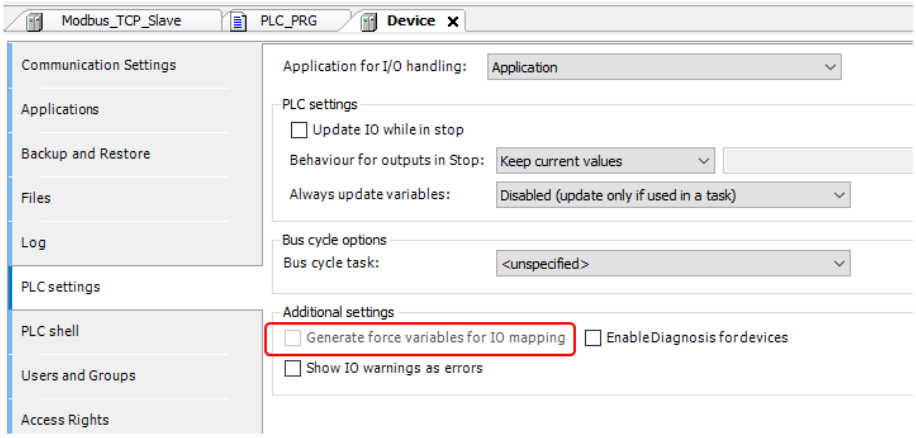


Рисунок 6.51 – Увімкнення генерації фіксованих змінних

Зміни вступають в дію після виконання компіляції програми. Дві змінні створюються для кожного каналу вводу / виводу відповідно до наступного синтаксису, у процесі якого пробіли в назві каналу замінюються підкресленням:

– <device name>_<channel name>_<IEC address>_Force типу BOOL для активації та дезактивації закріплення;

– <device name>_<channel name>_<IEC address>_Value типу даних каналу для визначення значення, яке необхідно примусово застосувати до каналу.

Ці змінні доступні в помічнику вводу в категорії Variables / IoConfig_Globals_Force_Variables та є можливість використовувати їх в CODESYS в об’єктах програмування, візуалізаціях, конфігурації символів тощо.

6.4 Створення графічного інтерфейсу користувача за допомогою CODESYS

6.4.1 Основи створення графічного екрану

Середовище CODESYS надає змогу створювати графічні екрани, які можна використовувати для керування, як локальними так і віддаленими пристроями. Графічні екрани можна завантажувати в спеціальні пристрої (графічні панелі оператора), або використовувати Web-браузери для керування пристроями через мережу Ethernet.

Після створення нового проєкту, до нього необхідно додати новий об’єкт візуалізації (рис. 6.52).

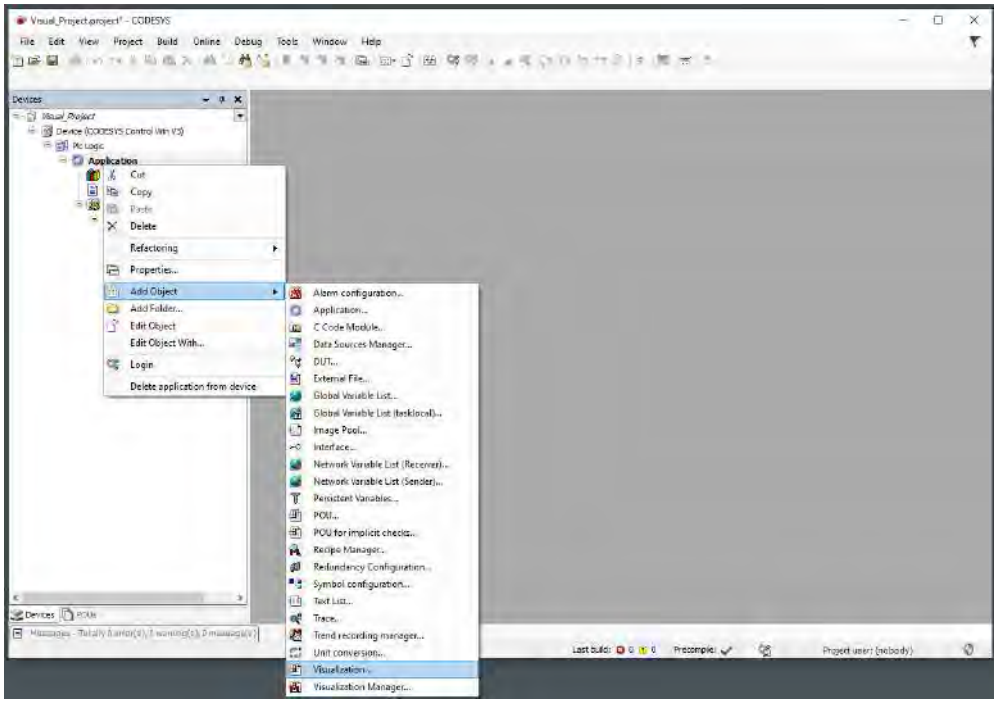


Рисунок 6.52 – Додавання панелі візуалізації до проєкту

При створенні екрану візуалізації необхідно задати його ім’я та, за потреби, обрати бібліотеку спеціальних символів (рис. 6.53).

В разі успішного створення нового об’єкту, в проєкт також буде додано ще два додаткових компонента (рис. 6.54):

– менеджер візуалізації;

– об’єкт таргет-візуалізації;

– об’єкт web-візуалізації.

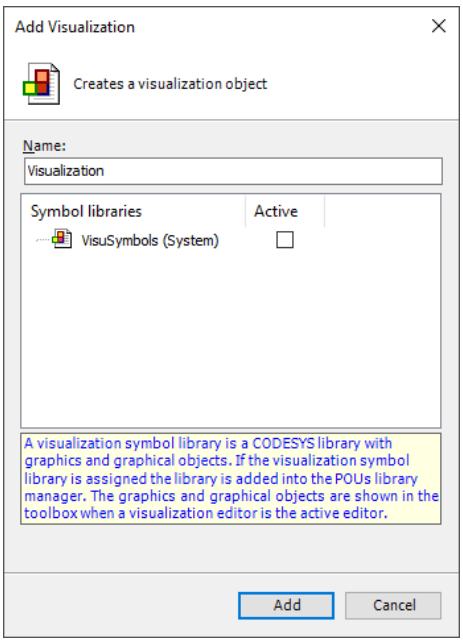


Рисунок 6.53 – Налаштування екрану візуалізації при створенні

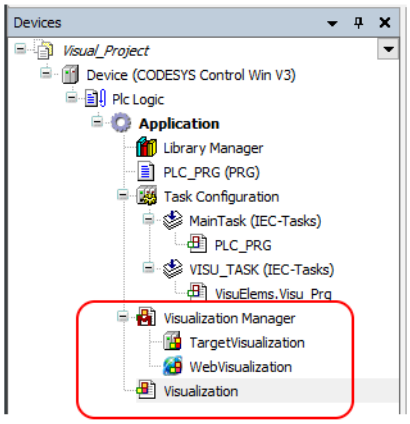


Рисунок 6.54 – Нові об’єкти у дереві проєкту

Таргет-візуалізація – це візуалізація, що відображається на цільовому пристрої (наприклад, панельному контролері). Кожен проєкт може містити лише одну таргет-візуалізацію.

Web-візуалізація – це візуалізація, що транслюється на задану IP-адресу. Вона може бути відкрита на будь-якому пристрої з браузером, який підтримує HTML5. Для запуску web-візуалізації цільовий пристрій повинен містити web-сервер. Кількість web-візуалізацій у проєкті не обмежена.

Віддалена таргет-візуалізація – це візуалізація, що відображається на ПК, підключеному до цільового пристрою. Цільовий пристрій повинен мати таргет-візуалізацію та ліцензію на віддалену таргет-візуалізацію.

Також існує сервісна візуалізація, що відображається у редакторі візуалізації CODESYS під час підключення до цільового пристрою із запущеним проєктом.

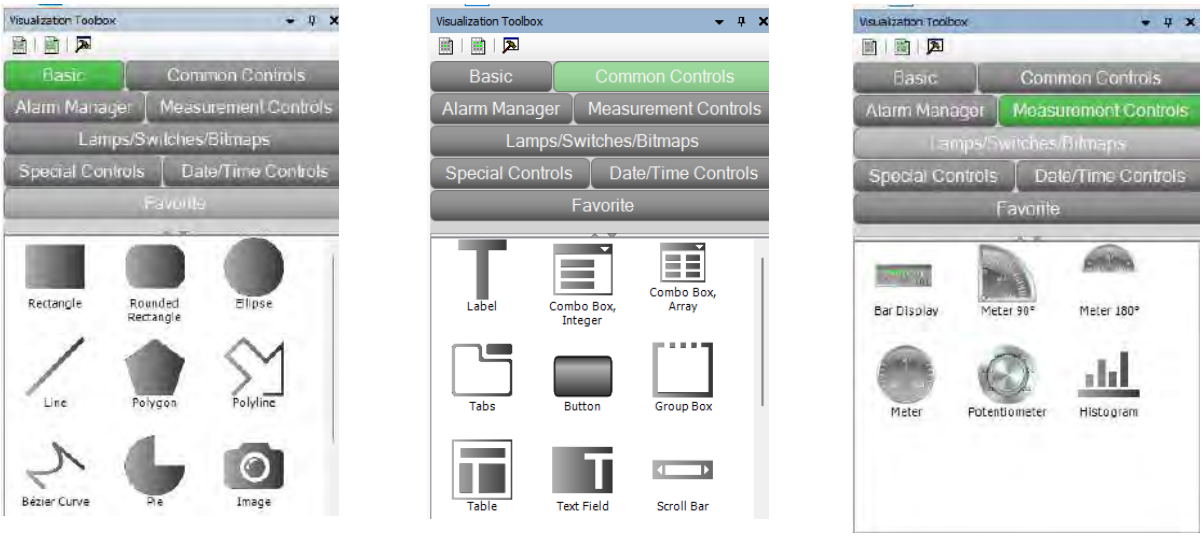
Основними елементами при створенні графічних екранів є:

– прямокутник, скруглений прямокутник та еліпс з набору базових компонентів (рис. 6.55, а);

– текстова мітка, таблиця, кнопка та ін., з набору стандартних елементів керування (рис. 6.55, б);

– аналогові прилади відображення інформації та різного типу гістограми, з набору елементів управління вимірами (рис. 6.55, в).

Також застосовуються об’ємні елементи візуалізації у вигляді кнопок, перемикачів, та лампових індикаторів з набору «Лампи та перемикачі».



а) б) в)

Рисунок 6.55 – Основні елементами для створення графічних екранів

Графічні елементи, в залежності від типу, можуть відображати звичайний текст, значення змінних, змінювати колір в залежності від стану змінної, впливати на вміст змінної в процесі взаємодії з елементом керування.

6.4.2 Властивості візуальних елементів

Розглянемо властивості елементів, що найчастіше застосовуються в проєктах. Для створення графічних інтерфейсів найчастіше використовуються елементи з базового набору, але принцип їх використання притаманний й іншим елементам.

Для прикладу візьмемо візуальний елемент «Прямокутник» (рис. 6.56).

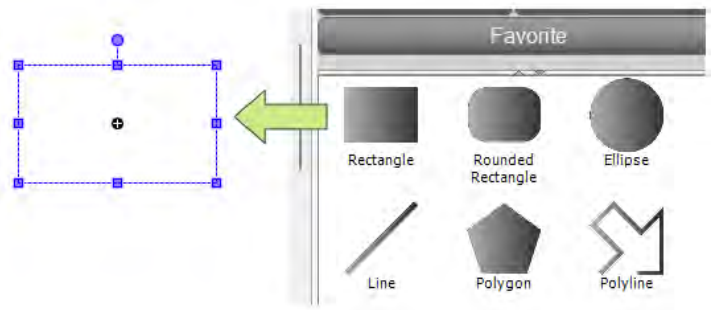


Рисунок 6.56 – Візуальний елемент «Прямокутник»

Даний елемент є універсальним і може застосовуватись для:

– виведення звичайного статичного тексту;

– виведення динамічного форматованого тексту з додаванням значень, що містяться в змінних проєкту;

– реалізації поведінки кнопки керування;

– реалізації функції індикатора із зміною кольору в залежності від стану змінної, що до нього прив’язана.

Даний елемент також може змінювати форму та, в залежності від налаштування, ставати скругленим прямокутником, або еліпсом.

В режимі відображення звичайного статичного тексту «Прямокутник» відображає текстовий рядок, що задано у властивості «Текст» редактора властивостей (рис. 6.57).

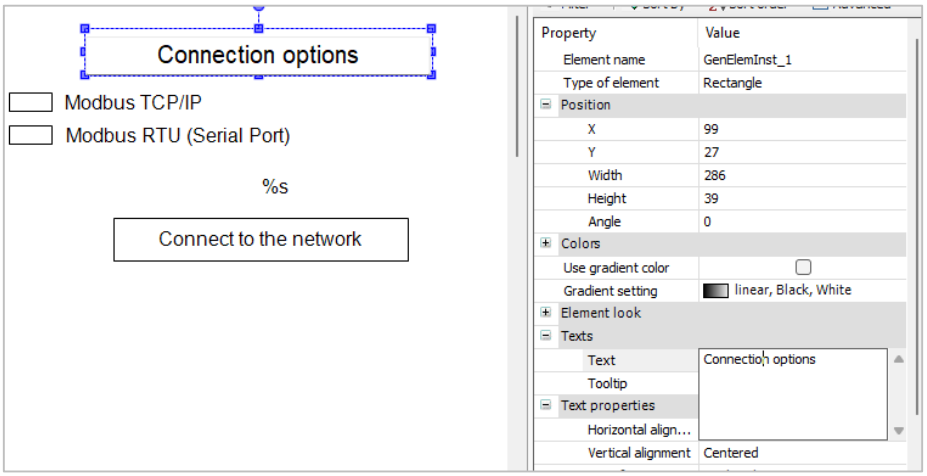


Рисунок 6.57 – Редагування параметрів текстової мітки

Для налаштування доступні всі стандартні властивості: розмір та тип шрифту, положення тексту в рамках елемента, колір тексту.

«Прямокутник» можна налаштувати для роботи в режимі індикатора стану. Для цього елемент потрібно прив’язати до булевої змінної та налаштувати поведінку в залежності від стану цієї змінної.

Для прикладу, розглянемо інтерфейс налаштування підключення, що зображено на рис. 6.58. В даному інтерфейсі, крім прямокутників в форматі текстових міток, використовуються два елементи, що планується налаштувати для роботи в режимі індикаторів.

Рисунок 6.58 – Приклад формування графічного інтерфейсу

Додамо в проєкт дві змінні «isModbusTCP» та «isModbusRTU» булевого типу (рис. 6.59).

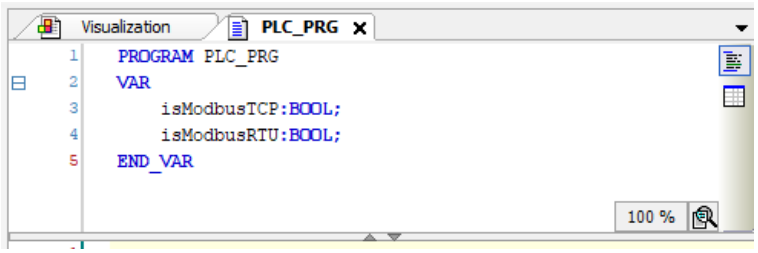


Рисунок 6.59 – Об’ява двох змінних «isModbusTCP» та «isModbusRTU»

В тексті програми присвоїмо їм значення:

```
isModbusTCP:=TRUE;
isModbusRTU:=FALSE;
```

На рис. 6.60 подано повний текст програми для демонстрації роботи зі змінними при створенні візуалізації.

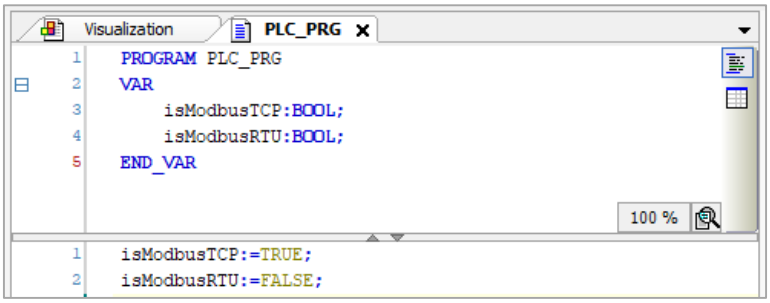


Рисунок 6.60 – Повний текст програми для демонстрації роботи зі змінними в процесі створення візуалізації

У режимі редагування інтерфейсу прив’яжемо змінні до графічних елементів (рис. 6.61).

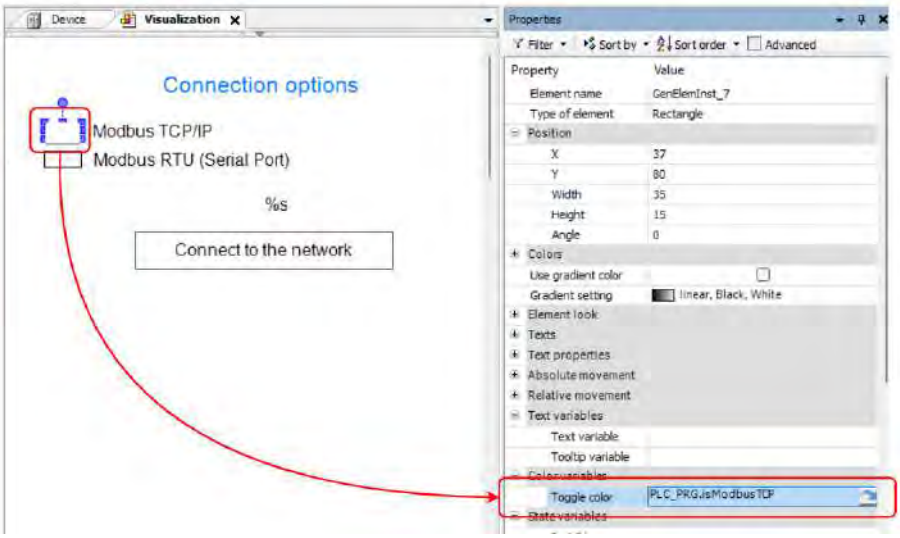


Рисунок 6.61 – Поле прив’язування змінної до графічного елементу

Змінну можна ввести вручну, або викликати вікно обирання її зі списку доступних у проєкті (рис. 6.62).

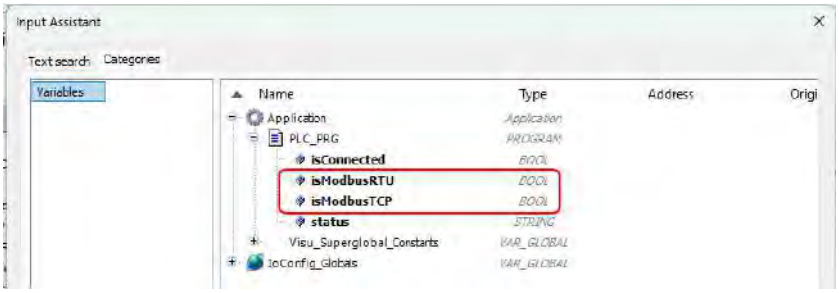


Рисунок 6.62 – Вибір змінної зі списку

В ручному режимі посилання на змінну записується в форматі:

PLC_PRG.isModbusTCP

Наступним кроком обираємо колір прямокутника для двох режимів: нормальний стан та режим тривоги (рис. 6.63). Нормальним режимом вважається випадок, коли значення змінної дорівнює FALSE, а режим тривоги – TRUE.

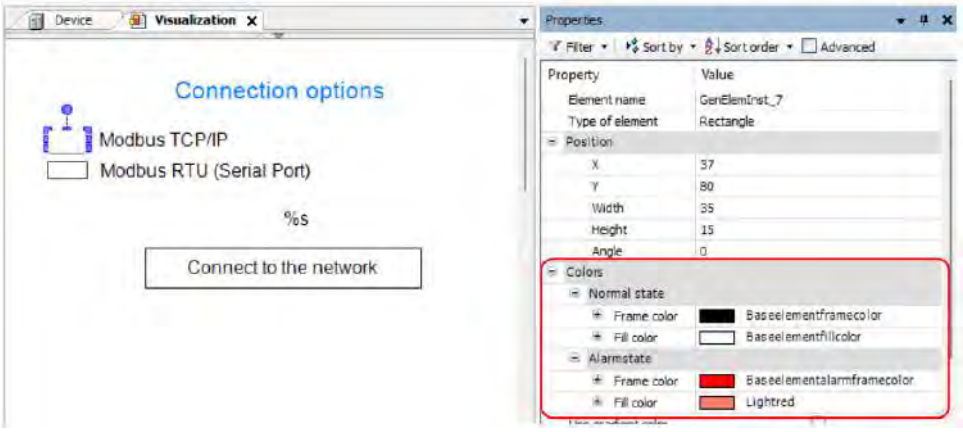


Рисунок 6.63 – Вибір кольору прямокутника для різних режимів

В нашому прикладі у випадку, коли змінна має значення FALSE, колір прямокутника буде білим, а коли TRUE – червоним.

Після компіляції та запуску програми, графічний інтерфейс матиме вигляд, як подано на рис. 6.64.

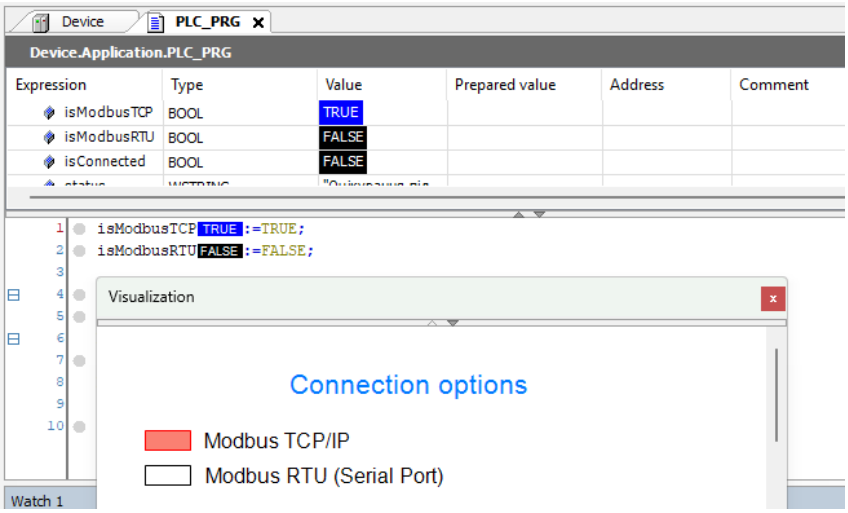


Рисунок 6.64 – Вигляд графічного інтерфейсу програми

З поданого рисунку, можна бачити стан змінних «isModbusTCP» і «isModbusRTU» та відповідність поведінки графічних елементів цим станам.

Візуальний компонент «Прямокутник» може виконувати функцію кнопки. Для демонстрації даної можливості, додамо в проєкт дві нових змінних «isConnected» типу Bool та текстову змінну «status» типу String для зберігання статусу з'єднання.

На форму додамо два нових графічних елементи «Прямокутник». Один буде виконувати функцію кнопки, інший – динамічної текстової мітки.

По натисканню на кнопку потрібно реалізувати можливість перемикання значення змінної «isConnected» з True на False і навпаки. Також необхідно міняти текст мітки в залежності від статусу з «Очікування підключення до мережі» на «Підключення до мережі» (рис. 6.65).

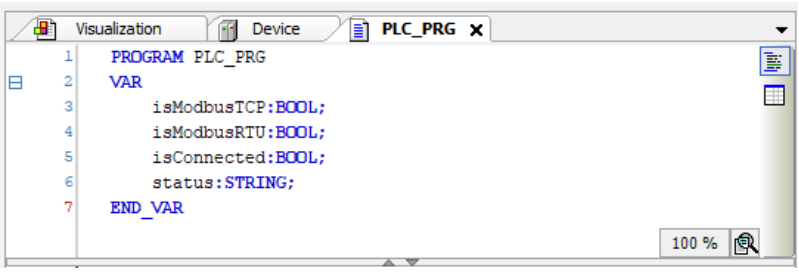


Рисунок 6.65 – Створення змінних для роботи кнопки

На рис. 6.66 подано приклад розташування нових візуальних компонентів.

Спочатку виконаємо налаштування кнопки. Для цього обираємо потрібний прямокутник та міняємо його властивість текст на «Підключитись до мережі». Для властивості «Конфігурація вводу» обираємо «Перемикання» та, за допомогою асистенту вводу, обираємо змінну «isConnected». Така властивість забезпечить нам можливість почергового перемикання статусу обраної змінної з TRUE на FALSE під час кожного натискання на кнопку.

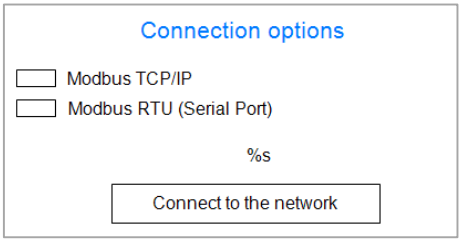


Рисунок 6.66 – Розташування нових візуальних компонентів

Можливо також обирання властивості «Натискання». В такому випадку, значення змінної «isConnected» буде змінюватись на TRUE, якщо кнопка натиснута, та повертатись назад до FALSE, якщо кнопку відпустити. Для рішення нашої задачі така поведінка не підходить.

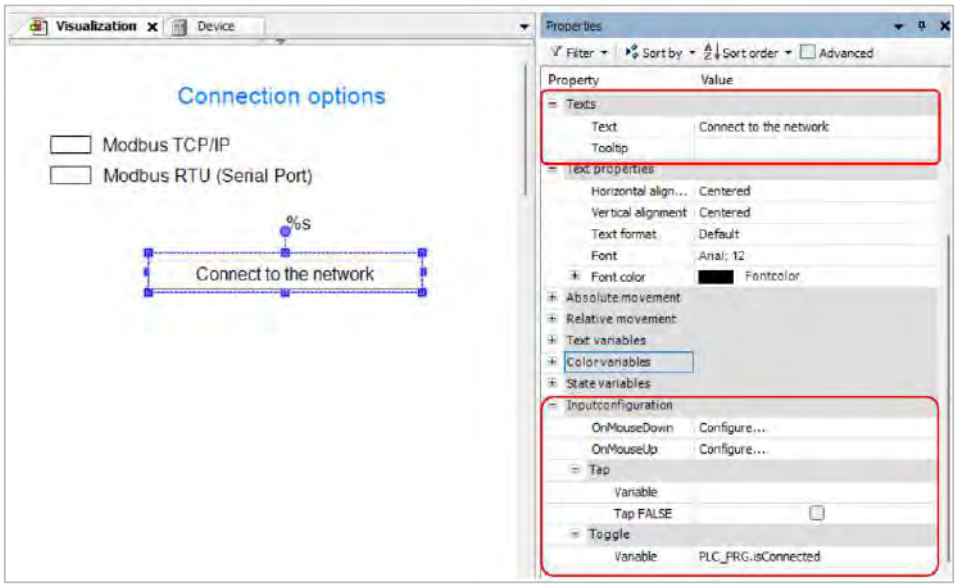


Рисунок 6.67 – Зміна властивостей кнопки

Для іншого компоненту, тексту статусу, задаємо динамічний текст за допомогою спеціального форматування «%s». Такий спосіб форматування текстового рядка дає можливість автоматично змінювати надпис в залежності від вмісту змінної. В якості такої змінної обираємо «status» (рис. 6.68).

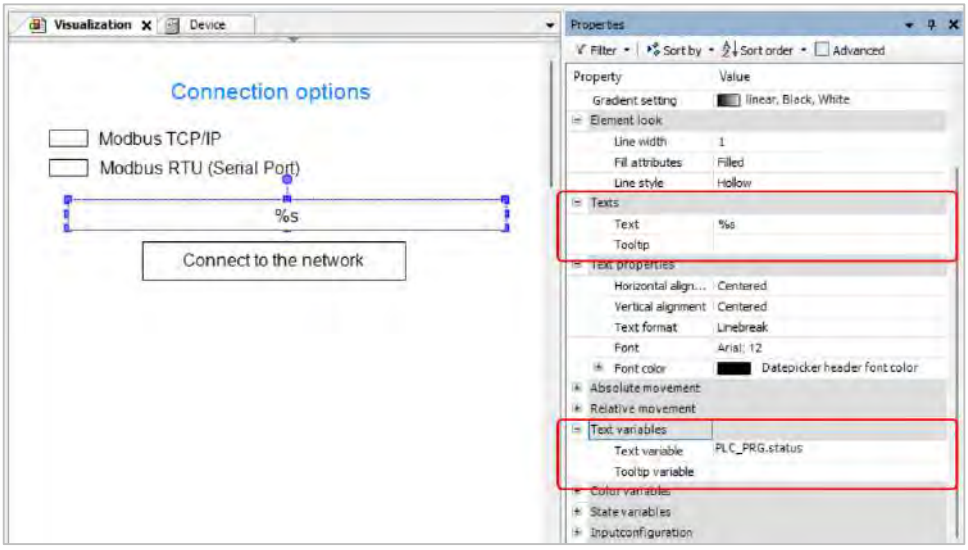


Рисунок 6.68 – Створення динамічної текстової мітки

Приклад роботи програми в режимі очікування натискання на кнопку подано на рис. 6.69.

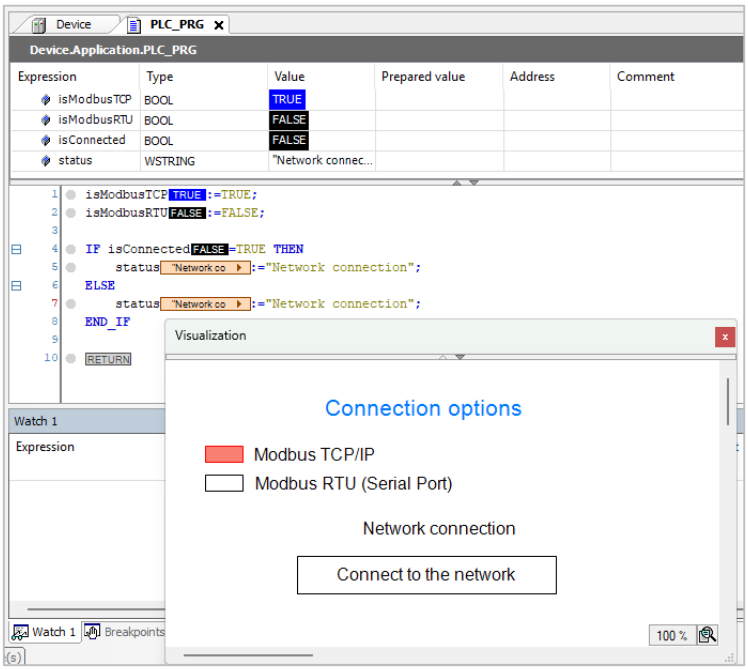


Рисунок 6.69 – Очікування натискання на кнопку

На рис. 6.70 подано графічний інтерфейс після натискання на кнопку «Підключитись до мережі».

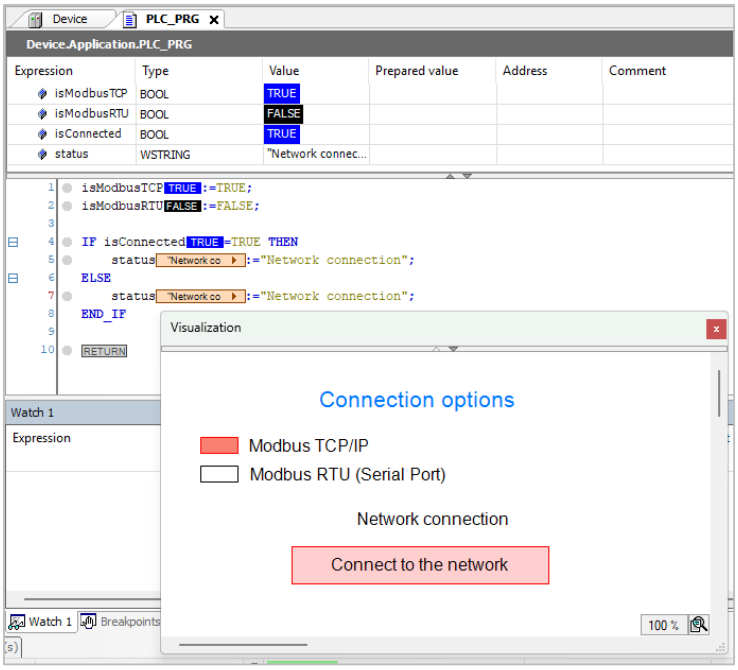


Рисунок 6.70 – Натискання на кнопку «Підключитись до мережі»

З поданих рисунків можна бачити, як змінюється вміст текстової мітки в залежності від стану змінної, до якої вона прив’язана. Також можна бачити реакцію кнопки на натискання.

На рис. 6.71 подано основний текст програми.

```
1  isModbusTCP:=TRUE;
2  isModbusRTU:=FALSE;
3
4  IF isConnected=TRUE THEN
5      status:="Network connection";
6  ELSE
7      status:="Network connection";
8  END_IF
```

Рисунок 6.71 – Основний текст програми

З поданого тексту можна бачити принцип заміни тексту в залежності від вмісту змінної «isConnect».

6.4.3 Форматування тексту

Властивість «Текст» в графічних компонентах може відображати значення різних типів змінних. Для правильного представлення інформації передбачені різні способи форматування.

Синтаксис заповнення властивості «Текст» подано на рис. 6.72.

Рисунок 6.72 – Синтаксис заповнення властивості «Текст»

На рис. 6.72 опція «Текст» – допоміжний текст (наприклад, назва та розмірність змінної).

Опція % – спеціальний символ, після якого вказується тип змінної, що відображається (перед типом можуть бути вказані налаштування формату виведення). Якщо потрібно відображати символ відсотка як текст, його слід записати як %%.

Точність – кількість символів, що виводяться після коми (для цілочисельних змінних), кількість символів, що виводяться (для рядкових змінних).

«Специфікатор формату виведення змінної» – символ формату виводу змінної, що відображається.

Список символів подано у таблиці 6.1.

Таблиця 6.1 – Список специфікаторів формату виведення змінної

Символ	Тип даних, що відображаються	Приклад використання		
		Фактичне значення	Форматування	Значення, що відображається
%d, i	Десяткове число	10	%d, i	10
%b	Двійкове число	10	%b	00000000 00001010
%o	Вісімкове число без знаку (без провідного нуля)	10	%o	12
%x	Шістнадцяткове число без знаку (без провідного нуля) для змінної з розміром до 32 біт	10	%x	a
%llx	Шістнадцяткове число без знаку (без провідного нуля) для змінної з розміром до 64 біт	16#4FFF_3FFF_2FFF_1FFF	%llx	4fff3fff2fff1fff
%u	Десяткове число без знаку	-10	%u	10
%c	Один символ з таблиці ASCII	16#21	%c	!
%s	Текстовий рядок	abcdef	%s %.3s	abcdef abc
%f	Дійсне число з плаваючою крапкою	10.1111	%f %.2f	10.111100 10.11
%e, %E	Дійсне число з плаваючою крапкою (змінна типу REAL)	7389056099	%.3e	7.389e+009
%g	Дійсне число з плаваючою крапкою (змінна типу REAL)	99990000 100000000 7389056099	%g %g %.3g	99990000.00000 1.000000e+008 7.389e+009
%t[формат часу]	Час	12:48:52	%t[HH:mm:ss]	12:48:52
%%	Символ відсотку	–	%%	%

Список доступних керуючих послідовностей, для змінних типу STRING/WSTRING, які використовуються під час роботи з текстом (перехід на новий рядок, повернення каретки і т. д.), подано в таблиці 6.2.

Таблиця 6.2 – Список керуючих послідовностей для текстових змінних

Символ	Результат використання / Відображуване значення
$$	$ (символ долару)
$’	‘ (апостроф)
$L	Перевід рядку
$N	Новий рядок
$R	Повернення каретки
$P	Нова сторінка
$T	Табуляція
$xx (xx – код символу в HEX)	Символ з таблиці ASCII для змінної типу String

Приклад використання керуючих послідовностей подано на рис. 6.73.

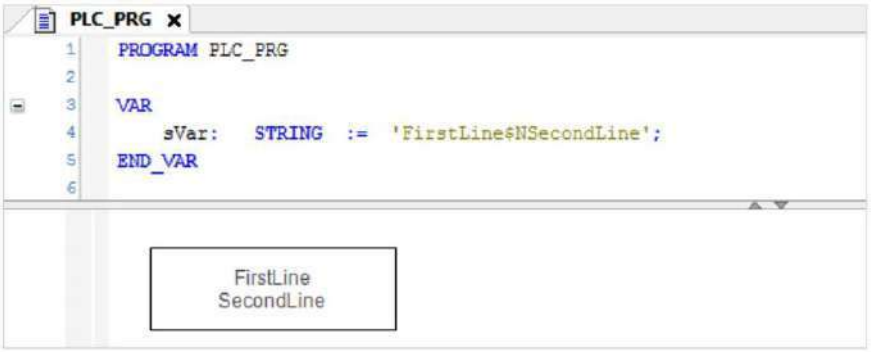


Рисунок 6.73 – Приклад використання керуючих послідовностей

На рис. 6.73 подано використання символу «Новий рядок». В результаті можна бачити, що в візуальному компоненті вміст змінної sVar займає два рядки.

Заповнювачі формату часу подані в таблиці 6.3.

Якщо, крім часу, необхідно відображати супровідний текст, слід укладати цей текст в одиночні лапки (наприклад, %t[dd 'days' hh 'hours']).

Таблиця 6.3 – Заповнювачі формату часу

Заповнювач	Значення, що відображається	Приклад відображення
ddd	Скорочена назва дня тижня	Fri (п’ятниця)
dddd	Повна назва дня тижня	Monday (понеділок)
ddddd	День тижня у вигляді числа	1 (понеділок), 7 (неділя),
MMM	Скорочена назва місяця	Feb (лютий)
MMMM	Повна назва місяця	February (лютий)
d	День у вигляді числа 1-31	8
dd	День у вигляді числа 01-31	08
M	Місяць у вигляді числа 1-12	8
MM	Місяць у вигляді числа 01-12	08
jjj	День в році з нулем попереду 001-366	253
y	Рік в форматі 0-99	8
yy	Рік в форматі 00-99	08
yyyy	Рік повністю	2023
HH	Час в форматі 0-24	08
hh	Час в форматі 0-12	08
m	Хвилини в форматі 0-59	8
mm	Хвилини в форматі 00-59	08
s	Секунди в форматі 0-59	8
ss	Секунди в форматі 00-59	08
ms	Мілісекунди в форматі 0-999	888
us	Мікросекунди в форматі 0-999	888
ns	Наносекунди в форматі 0-999	888
t	Ідентифікатор для 12-годинного формату: A (години < 12) і P (години > 12)	A (для 8 години) P (для 15 години)
tt	Ідентифікатор для 12-годинного формату: AM (години < 12) та PM (години > 12)	PM (для 15 години)

6.5 Контрольні питання

1. Яке призначення середовища програмування CODESYS у системах автоматизації?
2. Як створити новий проєкт у CODESYS?
3. Які можливості надає режим емуляції в середовищі CODESYS?
4. Що таке віртуальний контролер CODESYS і для чого його використовують?
5. Як реалізується обмін даними з пристроями за протоколом Modbus у CODESYS?

6. Які кроки потрібно виконати для створення та налаштування каналу Modbus у CODESYS?

7. Як налаштувати з’єднання з віртуальними приладами в CODESYS через Modbus?

8. Яке призначення змінних у проєкті CODESYS і як їх пов’язати з ресурсами вводу / виводу?

9. Як здійснюється налаштування пристроїв у проєкті CODESYS?

10. Що таке пряма адресація у CODESYS і коли її доцільно використовувати?

11. Як правильно пов’язати ресурси пристрою з існуючою змінною в проєкті?

12. Як створити нову змінну в CODESYS і прив’язати її до ресурсу пристрою?

13. Яким чином виконується фіксація адреси в карті ресурсів вводу/виводу?

14. Що таке неявні змінні і як вони застосовуються для прив’язки ресурсів у CODESYS?

15. Як створити графічний інтерфейс користувача в CODESYS та налаштувати властивості візуальних елементів?

16. З яких основних елементів складається головне вікно проєкту в середовищі CODESYS?

17. Наведіть приклад проєкта з двома змінними *x* і *y*, де *x* збільшується на 1 кожен цикл роботи ПЛК, а змінна *y* аналогічно зменшується.

18. З яких етапів складається створення майстра мережі та реалізація підлеглого?

19. Як виконати налаштування відповідних каналів для доступу до необхідних ресурсів підлеглого пристрою?

20. Поясніть принцип взаємодії віртуального ПК, як посередника між середовищем CODESYS, та віртуальним макетом, або пристроєм.

21. Назвіть основні правила роботи з відображенням входів і виходів пристрою на змінні в CODESYS.

22. У чому полягає відмінність Таргет-візуалізації та Web-візуалізації?

23. Назвіть основні елементи при створенні графічних екранів.

24. Які функції може виконувати візуальний компонент «Прямокутник»?

7 ПРИКЛАДИ УПРАВЛІННЯ ВІРТУАЛЬНИМИ ПРИСТРОЯМИ ЗА ДОПОМОГОЮ ПРОТОКОЛУ MODBUS

7.1 Віддалене управління світлофором

За допомогою середовища CODESYS створимо програму віддаленого управління світлофором. Перемикання сегментів пристрою буде виконуватись командами з використанням протоколу Modbus. Програма повинна мати візуальний інтерфейс з можливістю увімкнення або вимкнення роботи світлофора за допомогою перемикача. Ще однією функціональною можливістю даної програми буде переведення світлофора в режим очікування, коли періодично вмикається та вимикається жовтий сегмент пристрою.

Після створення нового проєкту додамо до нього новий об’єкт візуалізації та обираємо елементи керування віддаленим пристроєм з набору «Лампи та перемикачі» (рис. 7.1).

Рисунок 7.1 – Вибір органів керування віддаленим пристроєм

Розміщуємо візуальні елементи на формі, як подано на рис. 7.2.

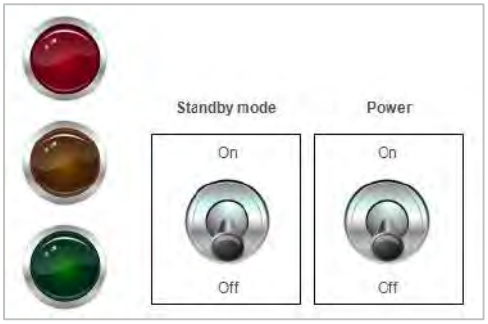


Рисунок 7.2 – Розміщення елементів на екрані візуалізації

Для кожного індикатора потрібно обрати робочій колір: червоний, жовтий та зелений, відповідно зверху до низу. Для цього виконується модифікація властивості «Зображення» для фону індикатора. На рис. 7.3 подано приклад обирання робочого кольору.

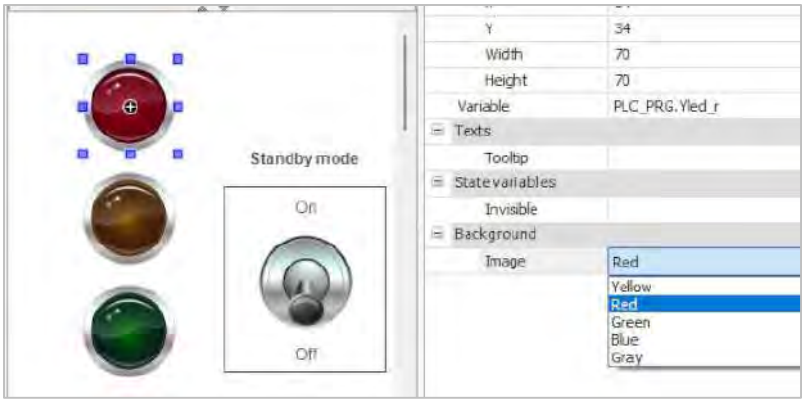


Рисунок 7.3 – Приклад обирання робочого кольору для індикаторів

Наступним кроком є прив’язка змінних до перемикачів. Для цього створимо дві змінні «isWork» та «isWait» типу BOOL (рис. 7.4).

```
6
7       isWork:BOOL;
8       isWait:BOOL;
9
```

Рисунок 7.4 – Змінні для роботи перемикачів

Змінна «isWork» призначена для опрацювання положення перемикача увімкнення живлення на світлофор. Коли isWork дорівнює TRUE, світлофор працює, коли isWork дорівнює FALSE, всі сегменти світлофору повинні бути вимкнені.

Якщо змінна «isWait» дорівнює TRUE, то світлофор переводиться в режим очікування. В такому режимі блимає жовтий сегмент пристрою, а інші сегменти не працюють. Якщо змінна «isWait» дорівнює FALSE, світлофор працює в звичайному режимі.

Прив’язка змінної до перемикача «Живлення» відбувається шляхом виклику асистенту введення ресурсів програми (рис. 7.5).

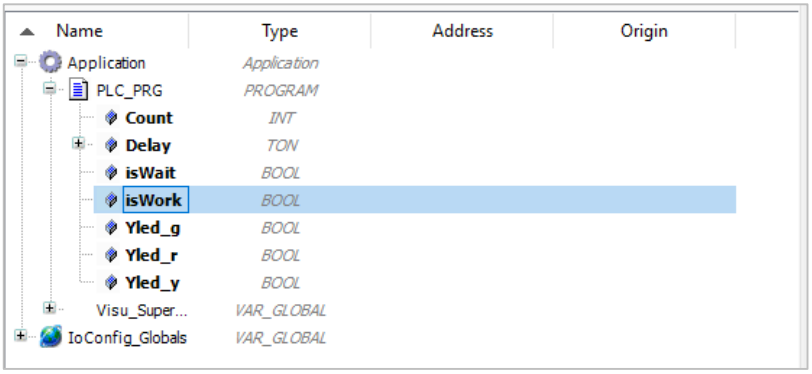


Рисунок 7.5 – Прив’язка змінної до перемикача

Так само прив’язується змінна «isWait» до перемикача «Режим очікування» (рис. 7.6).

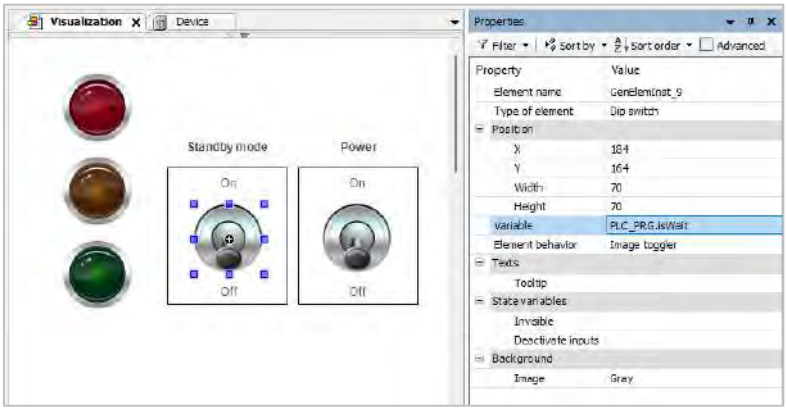


Рисунок 7.6 – Прив’язка змінної «isWait» до перемикача «Режим очікування»

Кожен з індикаторів також потрібно прив’язати до робочих змінних. В програмі передбачено три змінні: «Yled_r», «Yled_y» та «Yled_g». Кожна з цих змінних відповідає сегменту світлофору «Червоний», «Жовтий» та «Зелений» відповідно (рис. 7.7).

```
3       Yled_r:BOOL;
4       Yled_y:BOOL;
5       Yled_g:BOOL;
```

Рисунок 7.7 – Змінні для керування сегментами світлофору

Алгоритм роботи програми подано на рис. 7.8. На початку роботи виконується ініціалізація змінних та присвоєння їм початкових значень.

На кожному етапі роботи контролера, виконується затримка на 1 с, а після цього починається основний цикл роботи програми. Після перевірки увімкнення перемикача «Живлення» робиться висновок, чи продовжити роботу далі. Якщо

перемикач вимкнено, то всі сегменти світлофору вимикаються, а програма виходить з робочого циклу.

Якщо перемикач «Живлення» увімкнений, змінній isWork присвоюється значення TRUE та виконується перевірка стану перемикача «Режим очікування». Якщо цей перемикач також увімкнений, то програма переходить в режим почергового увімкнення та вимкнення тільки одного жовтого сегменту.

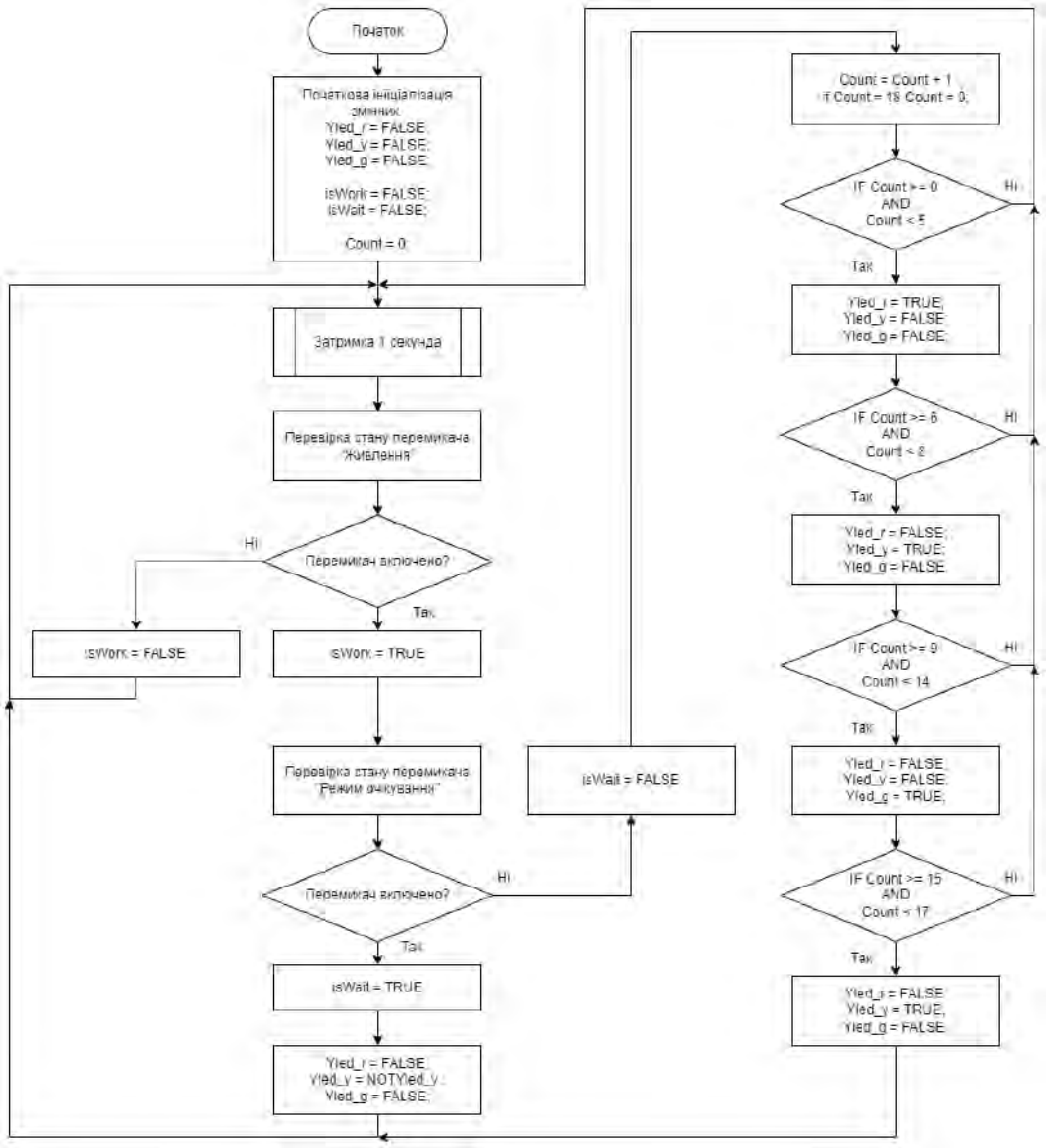


Рисунок 7.8 – Алгоритм роботи програми керування світлофором

Якщо перемикач «Режим очікування» вимкнено, змінна Count циклічно збільшується на одиницю. Збільшення відбувається до тих пір, поки воно не досягне максимального числа (18). В цьому разі змінна Count скидається до нуля.

Після кожної зміни вмісту Count виконується порівняння з одним із чотирьох шаблонів:

– якщо Count >= 0 та Count < 5, то вмикається червоний сегмент світлофору;

– якщо Count >= 6 та Count < 8, то вмикається жовтий сегмент світлофору;

– якщо Count >= 9 та Count < 14, то вмикається зелений сегмент світлофору;

– якщо Count >= 15 та Count < 17, то вмикається жовтий сегмент світлофору.

Приклад об’явлення змінних в коді програми подано на рис. 7.9.

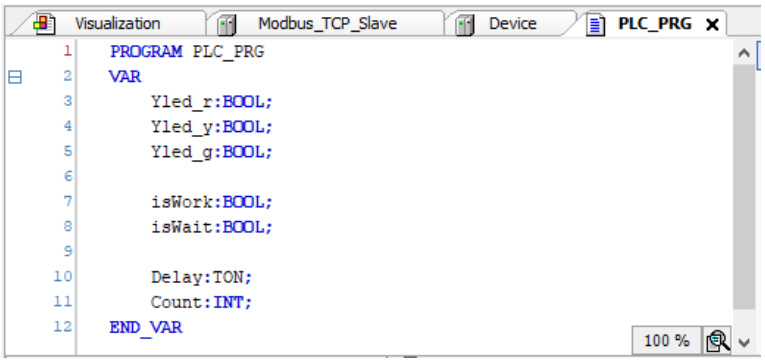


Рисунок 7.9 – Приклад об’явлення змінних в коді програми

На рис. 7.10 подано фрагмент програми реалізації затримки на 1 с.

```
Delay(IN:=TRUE, PT:=T#1S);
IF NOT(Delay.Q) THEN
    RETURN;
END_IF
Delay(IN:=FALSE);
```

Рисунок 7.10 – Фрагмент програми реалізації затримки на 1 с

В цьому фрагменті використовується об’єкт Delay типу TON, в який передається час затримки T#1S та дозвіл на виконання IN:=TRUE. Факт закінчення затримки можна дізнатися за станом виходу Q. Якщо він дорівнює TRUE, то виконується інша частина програми. В іншому випадку виконується вихід з головного циклу.

Фрагмент коду перевірки стану кнопки «Живлення» подано на рис. 7.11.

```
IF isWork = FALSE THEN
    Yled_r:=FALSE;
    Yled_y:=FALSE;
    Yled_g:=FALSE;
    RETURN;
END_IF
```

Рисунок 7.11 – Фрагмент коду перевірки стану кнопки «Живлення»

Якщо кнопка вимкнена, то програма далі не виконується, а всі сегменти світлофору вимикаються. Фрагмент коду перевірки стану кнопки «Режим очікування» подано на рис. 7.12.

```
14  IF isWait = TRUE THEN
15      Yled_r:=FALSE;
16      Yled_y:= NOT Yled_y;
17      Yled_g:=FALSE;
18      RETURN;
19  END_IF
```

Рисунок 7.12 – Фрагмент коду перевірки стану кнопки «Режим очікування»

З даного коду можна бачити, що рядок Yled_y:=NOT Yled_y з кожним викликом змінює значення змінної Yled_y на протилежний.

Нарощування значення змінної Count відбувається таки чином, як подано на рис. 7.13.

```
21  Count:=Count+1;
22  IF Count=18 THEN
23      Count:=0;
24  END_IF
25
```

Рисунок 7.13 – Нарощування значення змінної Count

Також, в даному фрагменті перевіряється на досягнення максимального значення та скидання Count в нуль.

Наступний фрагмент коду демонструє принцип порівняння значення змінної Count з шаблонами та увімкнення відповідного сегменту світлової колони (рис. 7.14).

```
26  IF Count >= 0 AND Count < 5 THEN
27      Yled_r:=TRUE;
28      Yled_y:=FALSE;
29      Yled_g:=FALSE;
30  ELSIF Count >= 6 AND Count < 8 THEN
31      Yled_r:=FALSE;
32      Yled_y:=TRUE;
33      Yled_g:=FALSE;
34  ELSIF Count >= 9 AND Count < 14 THEN
35      Yled_r:=FALSE;
36      Yled_y:=FALSE;
37      Yled_g:=TRUE;
38  ELSIF Count >= 15 AND Count < 17 THEN
39      Yled_r:=FALSE;
40      Yled_y:=TRUE;
41      Yled_g:=FALSE;
42  END_IF
```

Рисунок 7.14 – Увімкнення відповідного сегменту світлової колони

Виконаємо інтеграцію протоколу Modbus в наш проєкт. Для цього додамо до проєкту пристрій Ethernet, пристрій Modbus_TCP_Master та Modbus_TCP_Slave. Також створимо три канали Modbus для доступу до кожного сегмента (рис. 7.15).

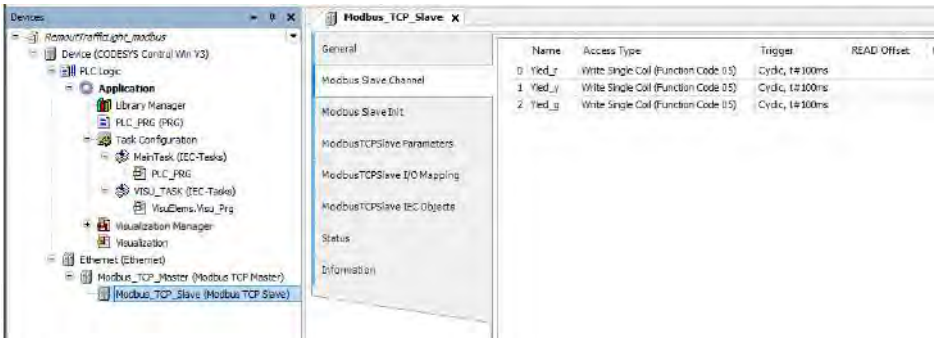


Рисунок 7.15 – Інтеграція протоколу Modbus в проєкт

Кожен канал відповідає сегменту світлової колони, тому назви каналів будуть відповідними: Yled_r, Yled_y та Yled_g. Кожен канал створюється аналогічно (рис. 7.16).

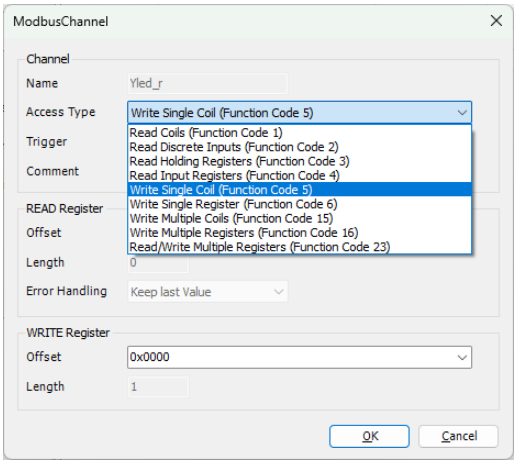


Рисунок 7.16 – Створення каналу Modbus

Управління сегментами відбуватиметься за допомогою функції 05 (Запис одного вихідного контакту) (рис. 7.17). Кожен канал має унікальне зміщення адреси відносно початкової: канал Yled_r – 0x0000, канал Yled_y – 0x0001 та канал Yled_g – 0x0002.

	Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offse
0	Yled_r	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0000
1	Yled_y	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0001
2	Yled_g	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0002

Рисунок 7.17 – Призначення адрес каналам Modbus

Після створення каналів виконуємо прив’язку змінних програми до ресурсів каналів Modbus (рис. 7.18).

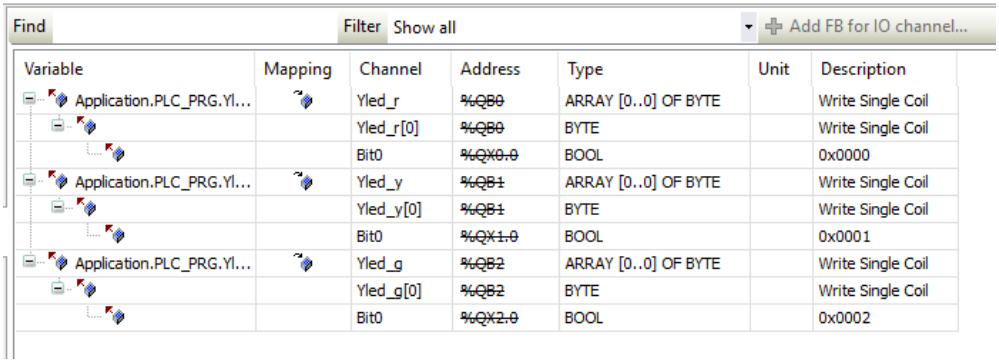


Рисунок 7.18 – Прив’язка змінних програми до ресурсів каналів Modbus

В результаті компіляції та запуску програми отримуємо наступний графічний інтерфейс (рис. 7.19).

Рисунок 7.19 – Графічний інтерфейс користувача

З поданого рисунку можна бачити, що світлофор знаходиться в робочому стані, горить червоний сигнал та кнопка увімкнення живлення перебуває в положенні «On». На рис. 7.20 подано стан каналів Modbus в режимі налагодження.

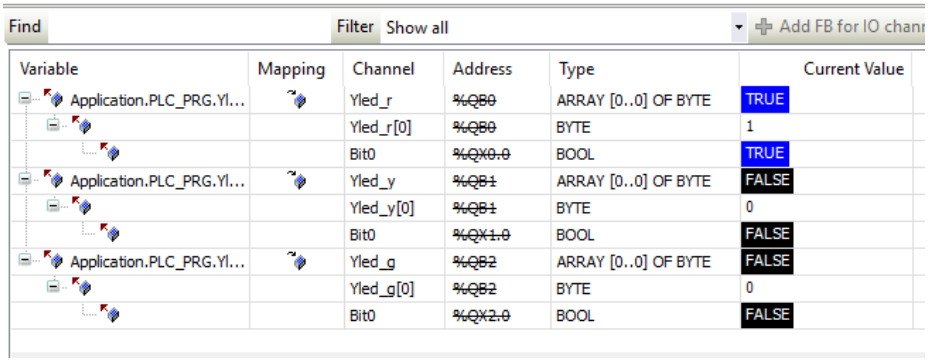


Рисунок 7.20 – Стан каналів Modbus в режимі налагодження

Можна бачити, що канал Yled_r та відповідна змінна знаходяться в стані «TRUE», тоді як інші – у стані «FALSE». Даний стан змінних транслюється за допомогою протоколу Modbus на віддалений пристрій, що призводить до увімкнення відповідного сегменту світлової колони, як подано на рис. 7.21.

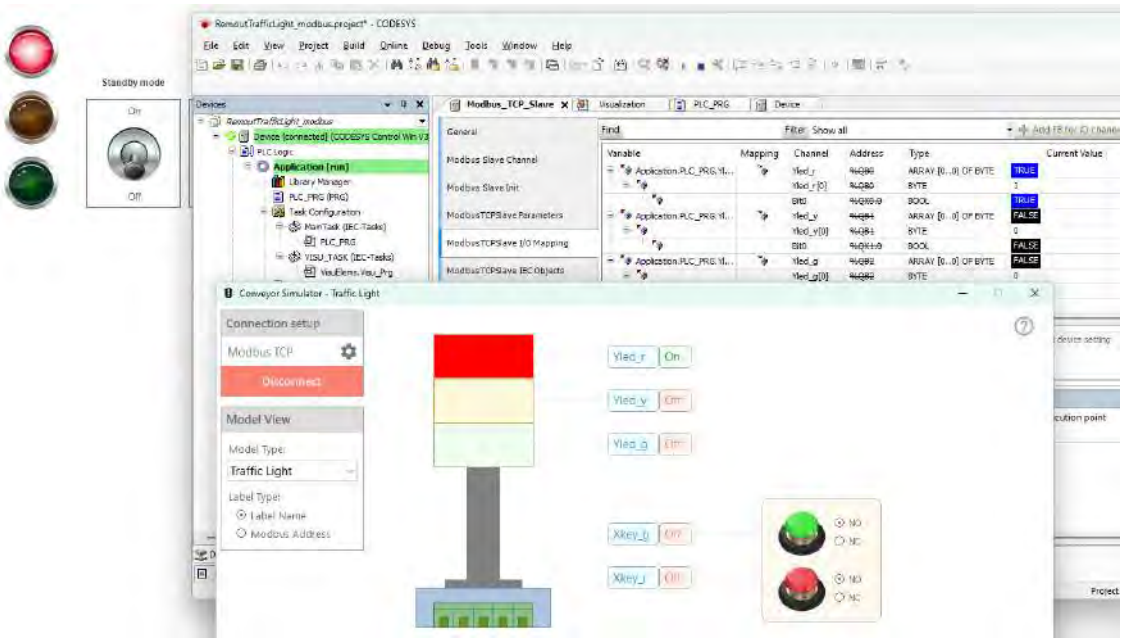


Рисунок 7.21 – Робота віддаленого пристрою через протокол Modbus

7.2 Автоматизація управління конвеєрною лінією

Опис віртуального макету «Промисловий конвеєр» надано в п. 5.2.3 даного навчального посібника. Для автоматизації конвеєра, потрібно реалізувати наступні функції:

– видача деталі по натисканню кнопки на віддаленій графічний панелі;

– моніторинг наявності деталей у всіх типах накопичувачів;

– автоматичне розподілення потоків деталей за принципом: парні перенаправляються в перший накопичувач, непарні пропускаються в кінцевий накопичувач;

– керування макетом за допомогою протоколу Modbus.

На рис. 7.21 подано графічний інтерфейс віртуального макета промислового конвеєра, на якому показані органи керування, що будуть задіяні в проєкті.

Рух конвеєрної стрічки забезпечує двигун YC_Moto. Він підключений до вихідних контактів з адресою Modbus 0x0064. Видача деталей відбувається

виштовхувачем YC_Stor1. Даний компонент підключений до вихідних контактів з адресою Modbus 0x0067. Напрям руху деталей визначає розподільник YC_Sp1. Важіль підключено до вихідних контактів з адресою Modbus 0x0065.

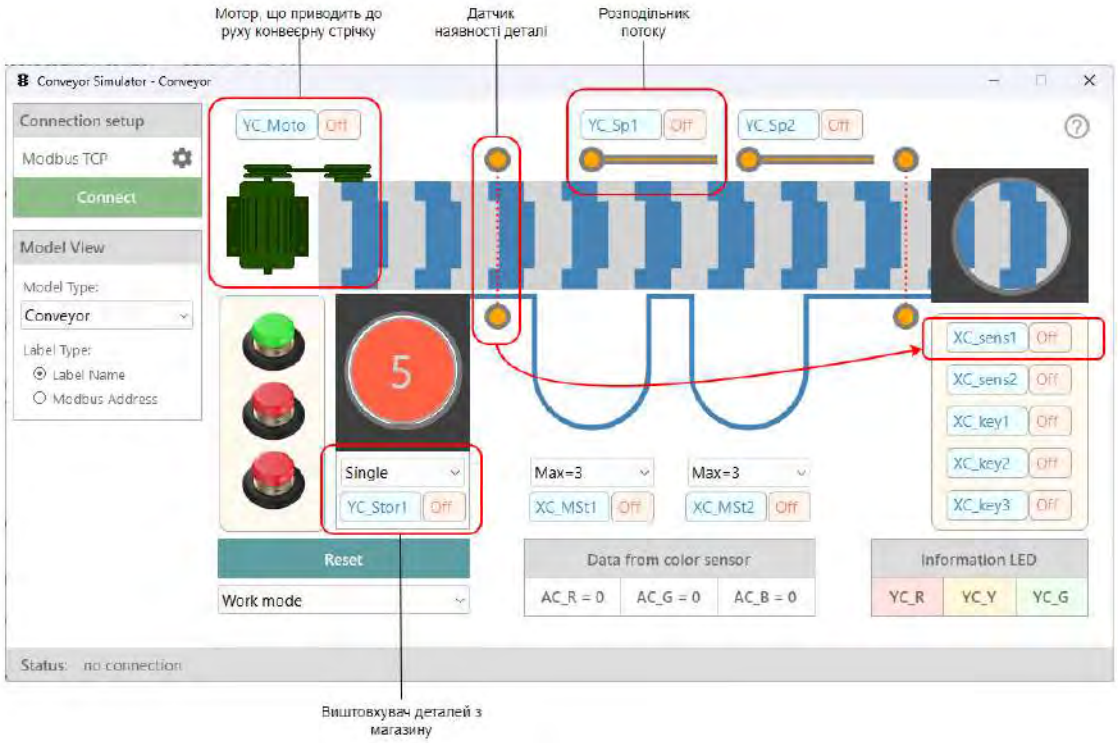


Рисунок 7.21 – Інтерфейс віртуального макета промислового конвеєра та органи керування

Контроль наявності деталей на конвеєрній стрічці виконується датчиком XC_sens1. Датчик підключено до вхідних контактів. Зміщення при адресації у функції читання стану вхідних контактів становить +0x0067.

За умовами завдання необхідно здійснювати віддалений моніторинг стану основних вузлів конвеєру:

– стан мотору, що рухає конвеєрну стрічку;

– положення розподільника потоку;

– кількість деталей, що залишилася в магазині, що живить конвеєрну лінію;

– кількість деталей в першому накопичувачі;

– кількість деталей в другому накопичувачі;

– кількість деталей в кінцевому накопичувачі.

На рис. 7.22 показані зазначені компоненти на інтерфейсі макету.

Стани мотору та розподільника потоку будуть визначатись шляхом використання функції Modbus для читання вихідних контактів. Зміщення адреси компонентів в адресному просторі Modbus +0x0064 та +0x0065 відповідно.

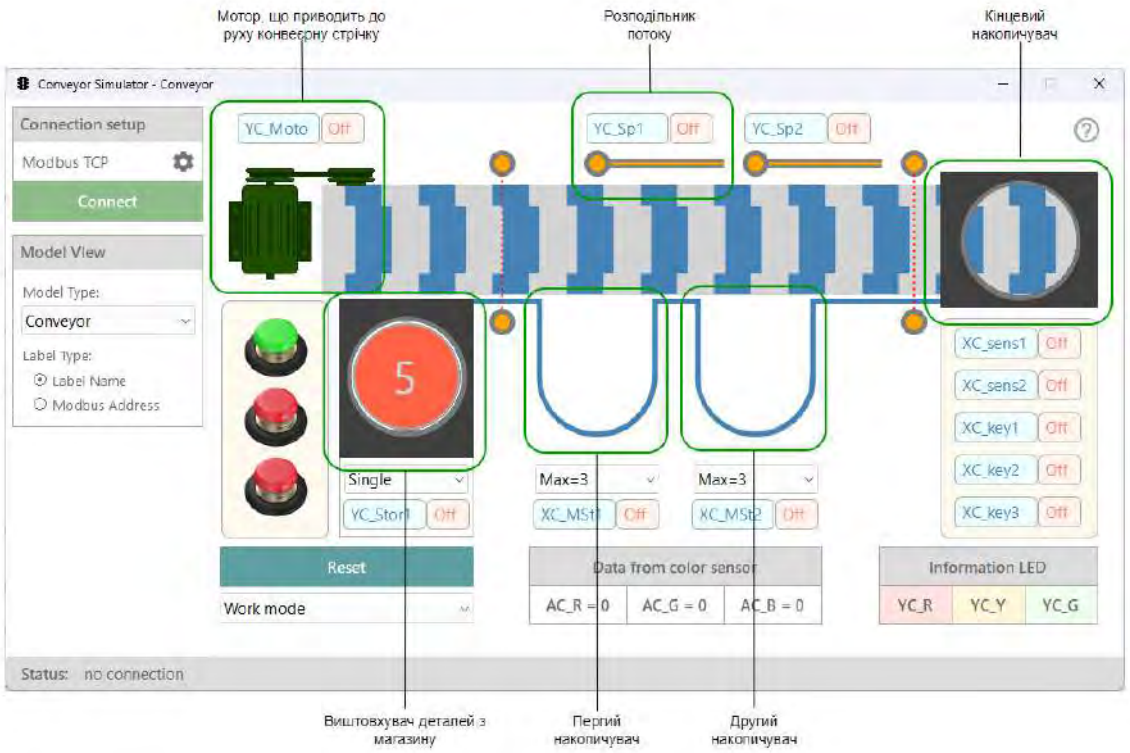


Рисунок 7.22 – Компоненти для моніторингу стану конвеєра

Кількість деталей в магазині та накопичувачах відображається в регістрах зберігання, починаючи з адреси 0x0064. Всього використовується чотири регістри.

Враховуючи вказаний розподіл пам’яті та адресний простір Modbus створимо необхідні канали в середовище CODESYS. На рис. 7.23 червоним виділені канали для управління макетом, а на рис. 7.24 зеленим – канали для моніторингу стану роботи конвеєрної лінії.

	Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset
0	YC_Magz	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0067
1	YC_Moto	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0064
2	Stor	Read Holding Registers (Function Code 03)	Cyclic, t#100ms	16#0064	4	Keep last Value	
3	monMoto	Read Coils (Function Code 01)	Cyclic, t#100ms	16#0064	1	Keep last Value	
4	monSp1	Read Coils (Function Code 01)	Cyclic, t#100ms	16#0065	1	Keep last Value	
5	XC_Sens1	Read Discrete Inputs (Function Code 02)	Cyclic, t#100ms	16#0067	1	Keep last Value	
6	YC_Sp1	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0065

Управління розподільником потоку
Читання даних з датчика наявності деталей на конвеєрі
Управління роботою мотору, що рухає конвеєрну стрічку
Управління штоком, що вишовхує деталі з магазину

Рисунок 7.23 – Канали для управління макетом

Рисунок 7.24 – Канали для моніторингу стану роботи конвеєрної лінії

Для функціонування програми необхідно створити набір змінних, що використовуватимуться для отримання або передачі даних за допомогою протоколу Modbus. На рис. 7.25 подано блок таких програмних змінних.

```
YC_Moto:BOOL;
YC_Magz:BOOL;

mon_YC_Moto:BOOL;
mon_YC_Sp1:BOOL;

Stor_1:WORD:=0;
Stor_2:WORD:=0;
Stor_3:WORD:=0;
Stor_4:WORD:=0;

YC_Sp1:BOOL;
XC_Sens1:BOOL;
isSens1:BOOL:=FALSE;
```

Рисунок 7.25 – Блок змінних програмного інтерфейсу

Змінна YC_Moto типу BOOL використовується для управління мотором конвеєрної ліні. Змінна YC_Magz типу BOOL використовується для управління виштовхувачем магазину деталей.

Змінна YC_Sp1 типу BOOL управляє роботою розподільника потоку деталей на конвеєрній стрічці. Змінна XC_Sens1 типу BOOL використовується для зберігання даних про стан датчика наявності деталей на лінії. Змінна isSens1 типу BOOL використовується в програмі для очікування, поки датчик наявності деталі повернеться в початковий стан.

Для моніторингу стану виробничої лінії передбачені наступні змінні. Змінна mon_YC_Moto типу BOOL використовується для зберігання інформації про стан двигуна конвеєра. В змінну mon_YC_Sp1 типу BOOL записується інформація про стан розподільника потоку.

Чотири змінних Stor_1, Stor_2, Stor_3 та Stor_4 типу WORD використовується для запису прочитаних даних про залишок деталей в магазині та в накопичувачах макету.

На рис. 7.26 можна бачити прив’язку програмних змінних до ресурсів каналів Modbus.

Find | Filter Show all | Add FB for IO channel... | Go to instance

Variable	Mapping	Channel	Address	Type	Unit	Description
Application.PLC_PRG.YC_Magz		YC_Magz	%QB0	ARRAY [0..0] OF BYTE		Write Single Coil
Application.PLC_PRG.YC_Moto		YC_Moto	%QB1	ARRAY [0..0] OF BYTE		Write Single Coil
		Stor	%IW0	ARRAY [0..3] OF WORD		Read Holding Registers
Application.PLC_PRG.Stor_1		Stor[0]	%IW0	WORD		0x0064
Application.PLC_PRG.Stor_2		Stor[1]	%IW1	WORD		0x0065
Application.PLC_PRG.Stor_3		Stor[2]	%IW2	WORD		0x0066
Application.PLC_PRG.Stor_4		Stor[3]	%IW3	WORD		0x0067
Application.PLC_PRG.mon_YC_Moto		monMoto	%IB8	ARRAY [0..0] OF BYTE		Read Coils
Application.PLC_PRG.mon_YC_Sp1		monSp1	%IB9	ARRAY [0..0] OF BYTE		Read Coils
Application.PLC_PRG.XC_Sens1		XC_Sens1	%IB10	ARRAY [0..0] OF BYTE		Read Discrete Inputs
Application.PLC_PRG.YC_Sp1		YC_Sp1	%QB2	ARRAY [0..0] OF BYTE		Write Single Coil

Рисунок 7.26 – Прив’язка програмних змінних до ресурсів каналів Modbus

За допомогою візуальних компонентів розробимо графічний інтерфейс панелі оператора (рис. 7.27).

Рисунок 7.27 – Графічний інтерфейс панелі оператора

На екрані оператора розташовані чотири різнокольорових панелі для відображення залишку деталей в магазині та в накопичувачах макета. Можна бачити, що для виведення числа використовується формат «%.2d», що у відповідності до принципу форматування текстових даних означає відображення двозначних чисел з нулем попереду.

Графічна панель має тумблер для увімкнення, або вимкнення роботи двигуна, що рухає конвеєрну стрічку.

Кнопка «Видати деталь» повинна бути пов’язана зі штоком, що виштовхує деталі з магазину.

Також в інтерфейсі передбачено два світлодіодних індикатора для контролю стану роботи двигуна та розподільника потоку, відповідно.

Для прив’язки графічного інтерфейсу до програми створено відповідний блок змінних (рис. 7.28).

Змінна isMoto типу BOOL пов’язана з тумблером в графічному інтерфейсі. Якщо тумблер увімкнений, значення змінної становить TRUE, якщо вимкнений – FALSE.

```
(*Visual Interface*)
isMoto:BOOL;
isMagazin:BOOL;

monMoto:BOOL;
monSp1:BOOL;

vStor_1:WORD:=0;
vStor_2:WORD:=0;
vStor_3:WORD:=0;
vStor_4:WORD:=0;
```

Рисунок 7.28 – Блок змінних графічного інтерфейсу

Змінна isMagazin типу BOOL пов’язана з кнопкою. Копка налаштована для роботи в режимі «Клавіша». Це означає, що в відповідній змінній з натисканням кнопки буде записано TRUE, а з відпусканням – FALSE.

Змінні monMoto та monSp1 типу BOOL пов’язані з візуальними компонентами у вигляді світлодіодних індикаторів. Індикатор «Стан лінії» (monMoto) має червоний колір, а індикатор «Стан сепаратора» (monSp1) – зелений.

Панелі відображення залишків деталей в магазині та накопичувачів пов’язані зі змінними vStor_1, vStor_2, vStor_3 та vStor_4 типу WORD.

Програма автоматизованого управління конвеєрною лінією складається з наступних блоків:

– блок управління роботою двигуна конвеєру:

```
IF isMoto = TRUE THEN
        YC_Moto := TRUE;
ELSE
        YC_Moto := FALSE;
END_IF
```

– блок управління виштовхувачем деталей із магазину:

```
IF isMagazin = TRUE THEN
        YC_Magz := TRUE;
ELSE
        YC_Magz := FALSE;
END_IF
```

– блок моніторингу стану конвеєрної лінії:

```
IF mon_YC_Moto = TRUE THEN
        monMoto := TRUE;
ELSE
        monMoto := FALSE;
END_IF
```

– блок моніторингу стану розподільника потоку деталей:

```
IF mon_YC_Sp1 = TRUE THEN
        monSp1 := TRUE;
ELSE
        monSp1 := FALSE;
END_IF
```

– блок моніторингу залишків деталей в магазині та накопичувачах:

```
vStor_1 := Stor_1;
vStor_2 := Stor_2;
vStor_3 := Stor_3;
vStor_4 := Stor_4;
```

– блок управління роботою розподільника потоку:

```
IF XC_Sens1 = TRUE THEN
        IF isSens1 = FALSE THEN
                YC_Sp1 := NOT YC_Sp1;
                isSens1 := TRUE;
```

```
        END_IF
ELSE
        isSens1 := FALSE;
END_IF
```

Останній блок виконує переключення сепаратору для кожної непарної деталі, що з’являється на лінії.

На рис. 7.29 подано приклад інтерфейсу програми та віртуального макета в процесі виконання технологічної програми.

Рисунок 7.29 – Приклад інтерфейсу програми та віртуального макета в процесі виконання технологічної програми

З поданого рисунку можна бачити, що виконавчі механізми макета відповідають положенню органам керування на панелі оператора. Інформаційні панелі відображають залишки деталей в магазині та накопичувачах.

Зелений індикатор показує, що важіль розподільника потоку знаходиться в положенні, що перекриває потік та цьому положенню відповідає зелений індикатор, що сигналізує про наявність активного сигналу на виході YC_Sp1.

7.3 Контрольні питання

1. Яке основне призначення використання протоколу Modbus для керування віртуальними пристроями?
2. У чому полягає суть завдання «Віддалене управління світлофором»?
3. Які компоненти входять до складу віртуального світлофора?
4. Які функції протоколу Modbus використовуються для перемикання сигналів світлофора?
5. Як реалізується послідовність сигналів світлофора за допомогою Modbus?
6. Які змінні використовуються для контролю стану світлофора у середовищі програмування?
7. Як забезпечується віддалений доступ до керування світлофором?
8. Які типові помилки можуть виникати при налаштуванні обміну даними з віртуальним світлофором?
9. У чому полягає мета автоматизації керування конвеєрною лінією?
10. Які елементи містить віртуальна модель конвеєрної лінії?
11. Як забезпечити синхронну взаємодію між різними вузлами конвеєра?
12. Які функції Modbus використовуються для запуску та зупинки конвеєра?
13. Як реалізується обробка сигналів від датчиків на конвеєрі за допомогою протоколу Modbus?
14. Як у CODESYS налаштувати логіку взаємодії між оператором і конвеєром через HMI?
15. Які переваги дає використання віртуального моделювання для навчання системам автоматизованого керування?
16. Наведіть приклад прив’язки змінних до перемикачів.
17. Які функції виконує об’єкт Delay типу TON?
18. Поясніть принцип інтеграції протоколу Modbus в проєкт.
19. Назвіть основні канали для управління макетом промислово конвеєру.
20. Назвіть основні канали для моніторингу стану роботи конвеєрної лінії.
21. Який вигляд має графічний інтерфейс панелі оператора для керування віртуальним приладом «Промисловий конвеєр»?
22. З яких основних блоків складається програма автоматизованого управління конвеєрною лінією?

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. S. Malakuti et al., Digital Twins for Industrial Applications. Definition, Business Values, Design Aspects, Standards and Use Cases [Електронний ресурс]: Industrial Internet Consortium, a program of Object Management Group, Inc. (“OMG”). Режим доступу: https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_Apps_White_Paper_2020-02-18.pdf. (дата звернення 08.09.2024).

2. Індустрія 5.0 (Industry 5.0) [Електронний ресурс]: IT-Enterprise. Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/industry-50. (дата звернення 27.09.2024).

3. Невлюдов І.Ш. Технологія програмування промислових контролерів в інтегрованому середовищі CODESYS: Навчальний посібник / І.Ш. Невлюдов, С.П. Новоселов, О.В. Сичова. Харків: ХНУРЕ, 2019 . 264 с.

4. ISO/IEC Organization. 2019. ISO/IEC 21823-1 Internet of things (IoT) – Interoperability for iot systems – Part 1: Framework.

5. Standardization for emerging technologies and innovations [Електронний ресурс]: JETI. Режим доступу: https://jtc1info.org/technology/advisory-groups/jeti/ (дата звернення 16.10.2024).

6. ISO/TC 184. Ad Hoc Group: Data Architecture of the Digital Twin. [Електронний ресурс]: International Organization for Standardization. Режим доступу: https://www.ththry.org/activities/2020/AdHocGroup_DigitalTwin_V1R8.pdf. (дата звернення 28.10.2024).

7. Digital Twins [Електронний ресурс]: IT-Enterprise. Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/cifrovoj-dvojnik-digital-twin. (дата звернення 13.11.2024).

8. Кафедра комп’ютерно-інтегрованих технологій, автоматизації та мехатроніки [Електронний ресурс] : офіційний веб-портал. – Режим доступу: https://tapr.nure.ua/. (дата звернення 25.11.2024)

9. Харківський національний університет радіоелектроніки [Електронний ресурс] : офіційний веб-портал. – Режим доступу: https://nure.ua/. (дата звернення 25.11.2024)

10. Новоселов С.П. Використання віртуальних лабораторних макетів для налагодження технологічних програм ПЛК / С.П. Новоселов, О.В. Сичова // Міжнародна науково-практична конференція «Інтелектуальні інформаційні

системи в управлінні проєктами та програмами», Коблево, 12-15 вересня 2023 р. Збірник праць. Харків: ХНУРЕ, 2023. С 12-15.

11. Невлюдов І. Ш. Застосування цифрових двійників технічних засобів автоматизації для розроблення програмно-технічних комплексів АСУ ТП: Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. Харків: Видавництво Іванченка І. С., 2023. 267 с. ISBN 978-617-8059-95-8. DOI: 10.30837/978-617-8059-95-8

12. M. Singh, E. Fuenmayor, E. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital Twin: Origin to Future," Applied System Innovation, vol. 4 (2), p. 36, 2021. DOI: 10.3390/asi4020036.

13. Road Vehicles-Controller Area Network (CAN) Part 1: Data Link Layer and Physical Signalling, Int. Org. Standard., Standard ISO 11898-1:2015, Dec. 2015.

14. A. Buscemi, I. Turcanu, G. Castignani, R. Crunelle, and T. Engel, "Poster: A methodology for semi-automated CAN bus reverse engineering," in Proc. 13th IEEE Veh. Netw. Conf. (VNC), Nov. 2021, pp. 125–126.

15. Modbus application protocol specification V1.1b3, 2012. [Електронний ресурс] :Modbus. Режим доступу: https://www.modbus.org/docs/Modbus_ Application_Protocol_V1_1b3.pdf. (дата звернення 23.12.2025).

16. Захищений модуль адаптера USB-RS485 [Електронний ресурс] : Arduino.ua. Режим доступу: https://arduino.ua/prod3168-zashhishhyonnii-modyl-adaptera-usb-rs485-na-ft232 (дата звернення 15.01.2025).

17. Перетворювач RS-485 на USB (CH340) [Електронний ресурс] : Arduino.ua. Режим доступу:https://arduino.ua/prod2773-preobrazovatel-rs-485-na-usb-ch340 (дата звернення 15.01.2025).

18. Невлюдов І.Ш. Node-RED та технологія промислового Інтернету речей: Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. Харків: Видавництво Іванченка І. С., 2024. 204 с. ISBN 978-617-8332-58-7. DOI: 10.30837/978-617-8332-58-7.

19. Details of the Administration Shell. Federal Ministry for Economic Affairs and Energy (BMWi). ZVEI & Plattform Indutrie4.0. Online: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details-of-theAsset-Administration-Shell-Part1.html.

20. D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," CIRP Journal of Manufacturing Science and Technology, vol. 29, pp. 36–52, 2020. doi: 10.1016/j.cirpj.2020.02.002.

21. M. Liu, S. Fang, H. Dong, and C. Xu, “Review of digital twin about concepts, tech-nologies, and industrial applications,” Journal of Manufacturing Systems, 2020. doi: 10.1016/j.jmsy. 2020.06.017.

22. “Digital Twin Cities: Framework and Global Practices”, World Economic Forum. Report April 2022. Available: https://www3.weforum.org/docs/ WEF_Global_Digi-tal_Twin_ Cities_Framework_and_Practice_2022.pdf

23. P. Tangtrustham, M. Okabe and Y. Takayanagi, "Industrial Control Product(s) with Virtualizing Technology Appropriate for CPS Platform," 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Chiang Mai, Thailand, 2020, pp. 13-18, DOI: 10.23919/SICE48898.2020.9240412.

Навчальне видання

НЕВЛЮДОВ Ігор Шакирович
НОВОСЕЛОВ Сергій Павлович
СИЧОВА Оксана Володимирівна
ТЕСЛЮК Сергій Ігорович

ПРОМИСЛОВІ МЕРЕЖІ ТА КОМПОНЕНТИ АСУТП

Навчальний посібник

Формат 60х84 1/16.
Ум. друк. арк. 14,1. Наклад 200 прим. Зам 30-06.

Видавництво
ФОП Іванченко І.С.
пр. Тракторобудівників, 89-а/62, м. Харків, 61135
тел.:+38(050/093) 40-243-50
Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготовників та розповсюджувачів видавничої продукції ДК № 4388 від 15.08.2012 р.
www.monograf.com.ua