МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

І.Г. Лисаченко, М.О. Подустов,
В.О. Лобойко, О. Г. Шутинський, А.К. Бабіченко

ПРОМИСЛОВІ МЕРЕЖІ: ТЕОРІЯ І ПРАКТИКА ЗАСТОСУВАННЯ ПРОТОКОЛІВ ТА ІНТЕРФЕЙСІВ

Навчальний посібник

Затверджено
редакційно-видавничою
радою університету,
протокол №1 від 03.02.2016 р.

Харків
Підручник НТУ «ХПІ»
2016

УДК 681.51
ББК 32.973.202
П88

Авторський колектив:
І.Г. Лисаченко, М.О. Подустов,
В.О. Лобойко, О. Г. Шутинський, А.К. Бабіченко

Рецензенти:
О.М. Рассоха, канд. техн. наук, проф. НТУ «ХПІ»; *В.Є. Корсун*, канд. техн. наук, доц. ХНУБА

П88 **Промислові мережі: теорія і практика застосування протоколів та інтерфейсів:** Навч. посібник / Лисаченко І.Г., Подустов М.О., Лобойко В.О., Шутинський О. Г., Бабіченко А.К. – Х.: Вид-во «Підручник НТУ "ХПІ"», 2016. – 176 с.
ISBN 978-617-687-064-7

У посібнику надано стислу характеристику та порядок застосування послідовних інтерфейсів, які є апаратною основою промислових мереж у складі АСУ ТП. Описано протоколи промислових мереж та наведено практичні рекомендації щодо їх застосуван-ня. На прикладі середовища *CoDeSys* показано програмну реалізацію протоколів *ModBus, OWEN* та *DCON* в промислових контролерах *ПЛК150 ОВЕН* (Україна). Запропоновано методику розроблення ЛМІ на основі РСУ.

Призначений для студентів спеціальності «Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання.

Іл. 75. Табл. 31. Бібліогр. 23 назв.

УДК 681.51
ББК 32.973.202

ISBN 978-617-687-064-7

ЗМІСТ

ВСТУП

Цей навчальний посібник розроблено для засвоєння загальних відомостей про мережі збору та обміну даними, які входять до структури АСУ ТП і В. Він призначений для студентів денної та заочної форм навчання за спеціальністю «*Автоматизація та комп'ютерно-інтегровані технології*». Посібник складається з чотирьох розділів, в яких послідовно йде розкриття базових понять про інформаційні та комунікаційні мережі.

У першому розділі наводяться відомості про побудову та основні принципи роботи інформаційних та комунікаційних мереж, а саме архітектуру та топологію мереж, методи кодування джерел та каналів, також розглянуто порядок доступу до каналів зв'язку.

Другий розділ присвячений питанням побудови та використання послідовних інтерфейсів, які є апаратною основою комунікаційних мереж, що входять до складу комп'ютеризованих систем управління технологічними процесами. Введено поняття «промислові мережі» та наведені їх основні характеристики, а також переваги та недоліки порівняно із звичайними комп'ютерними мережами. За допомогою використання прикладного програмного забезпечення та прикладів, які наведені в розділі, студенти можуть закріпити теоретичні знання про принципи дії основних послідовних інтерфейсів – *RS-232* та *RS-485*. Також дано пояснення концепції побудови еталонної моделі взаємодії відкритих систем та усіх її рівнів щодо застосування в промислових мережах.

У третьому розділі розглянуто протоколи промислових мереж, які базуються на використанні послідовних інтерфейсів *RS-232* та *RS-485*. Це протоколи *ModBus*, *OWEN* та *DCON*. Для цих протоколів наведені приклади, які демонструють принципи формування кадрів та аналіз їхнього вмісту. При цьому використані реальні апаратні засоби промислових мереж: промислові контролери, операторські панелі, конвертери інтерфейсів тощо.

У четвертому розділі на прикладі типового технологічного об'єкта розроблена розподілена система управління (РСУ) технологічним процесом з людино-машиним інтерфейсом (ЛМІ) у вигляді комп'ютерного практикуму. Виконання цього практикуму дозволяє студентам більш ефективно засвоїти в подальшому теорію та практику застосування промислових мереж та самостійно розробляти РСУ з ЛМІ.

РОЗДІЛ 1.
ОСНОВИ ПОБУДОВИ СИСТЕМ ОБМІНУ ДАНИМИ У СКЛАДІ АСУ ТП і В

Метою цього розділу є підготовка до засвоєння більш спеціалізованої інформації, яка буде наведена у наступних розділах.

Так, завершивши вивчення цього розділу, ви зможете:

- отримати уявлення про мережі і системи обміну даними та про її складові елементи;
- пояснити основи бінарної та 16-річної систем числення;
- описати чинники, які впливають на швидкість передачі даних та якість каналів зв'язку:
  - ширина смуги перепускання;
  - відношення сигнал/шум;
  - перепускна здатність;
  - частота появи похибок;
- пояснити призначення основних компонентів комунікаційної мережі;
- описати комунікаційні режими системи обміну даними;
- описати формат повідомлень та процедури виявлення помилок в асинхронних та синхронних системах обміну даними;
- перерахувати та пояснити порядок використання найбільш поширених кодів:
  - код Бодо та Морзе;
  - *ASCII*;
  - код Грея;
  - двійково-десятковий код (*BCD*);
- пояснювати методи доступу до фізичних каналів мереж;
- пояснювати принцип взаємодії багаторівневих систем зв'язку;
- отримати уявлення про концепцію побудови еталонної моделі взаємозв'язку відкритих систем;
- отримати загальне уявлення про поняття «промислові мережі», а також про їх переваги та недоліки порівняно з комп'ютерними мережами.

1.1. Загальна характеристика системи обміну даними

Системи, які здатні створювати, сприймати, відображати, зберігати та обробляти інформацію, називаються інформаційними системами. З цих позицій мережі АСУ ТП і В є інформаційними системами, а з урахуванням їхньої специфіки застосування – інформаційними вимірювально-управляючими системами (ІВУС). При цьому вказані системи не обмежуються наявністю всього одного каналу зв'язку, тобто частіше вони є багатоканальними. Багатоканальна система передачі інформації забезпечує одночасну і взаємно незалежну передачу повідомлень від багатьох джерел по одній загальній лінії зв'язку. Це досягається за рахунок використання додаткового обладнання. До складу ІВУС входять так звані *вузли*. Вузол зв'язку (інформаційний вузол) є достатньо складним компонентом системи, оскільки, окрім багатоканальної передачі (прийому) інформації, він забезпечує:

- вибір найкоротшого шляху між джерелом та одержувачем повідомлення (маршрутизація);
- дотримання системи пріоритетів (доступ до каналу зв'язку);
- накопичення і зберігання інформації за відсутності вільних каналів передачі (буферизація даних);
- комп'ютеризоване управління усіма перерахованими функціями в автоматичному режимі.

На цей час вже остаточно сформувалась як прикладна наука *теорія інформації*. Вочевидь, що предметом вивчення цієї теорії є термін «*інформація*». Зазвичай під цим поняттям мають на увазі нові відомості про навколишній світ шляхом взаємодії з ним, пристосування до нього та зміненя його в процесі пристосування. В теорії інформації [1] наведено визначення терміна «*інформація*».

Інформація це насамперед відомості, які повинні бути використані для вирішення завдань вимірювання, контролю та управління. Саме ІВУС належать до таких систем, які вирішують завдання вимірювання та управління у складі АСУ ТП і В. Зауважимо, що необхідно розрізнювати поняття «інформація» та «повідомлення». Повідомлення – це форма подання ін-

формації, наприклад покази вимірювального пристрою з підключеним до нього термометром опору, керуючі команди тощо.

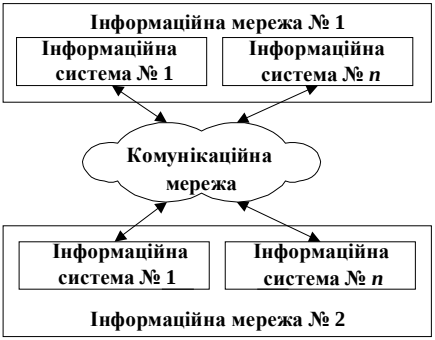


Рисунок 1.1 – Побудова та взаємозв'язок комунікаційної та інформаційної мереж

Таким чином, можна ввести такий термін – «інформаційна мережа», яка є сукупністю інформаційних вузлів, з'єднаних лініями зв'язку, яка є суб'єктом інформаційного простору. Взагалі, «мережа» – це термін, який є сукупністю пристроїв передавання, приймання та зберігання даних, а також засобів зв'язку між ними. Розрізняють мережі двох типів: *комунікаційні* та *інформаційні* (обчислювальні). Міжнародна організація стандартизації (*ISO – International Organization for Standardization*, http://www.iso.org) визначила *обчислювальну мережу* як систему для послідовної біт-орієнтованої передачі інформації між зв'язаними один з одним незалежними пристроями [2]. На рис. 1.1 показано структуру та взаємодію інформаційних та комунікаційних мереж.

Інформаційна мережа (*обчислювальна мережа*) – це сукупність інформаційних вузлів, з'єднаних лініями зв'язку, яка призначена для зберігання інформації та складається з інформаційних систем. Під інформаційною системою слід розуміти систему, яка є постачальником або споживачем інформації.

У свою чергу, *комунікаційна мережа* – це сукупність каналів зв'язку та термінальних пристроїв, які призначені для перетворення та передавання даних по різних фізичних середовищах.

В інформаційних та комунікаційних мережах циркулюють дискретні та безперервні повідомлення, які поділяються за характером змінювання у часі та за множиною.

Безперервне повідомлення у часі відображається безперервною функцією часу. Але й безперервні повідомлення можна передавати дискрет-

ними методами. Для цього безперервний сигнал (повідомлення) піддають дискретизації в часі та квантуванню за рівнем. На приймальній стороні виконується відновлення безперервної функції за дискретними відліками. Але ці операції потребують використання додаткового обладнання: аналого-цифрових та цифро-аналогових перетворювачів. *Дискретне повідомлення у часі* є послідовністю окремих символів, які передаються у визначені моменти часу. Для перетворення дискретного повідомлення в сигнал необхідно виконати операцію кодування повідомлення, при якому підвищується швидкість та перешкодостійкість інформації, що передається.

Поділ повідомлень на дискретності за множиною та у часі не пов'язані один з одним. На рис. 1.2 показано усі можливі типи повідомлень:

- безперервні за множиною та у часі, звичайні безперервні (рис. 1.2, *а*);
- безперервні за множиною та дискретні у часі (рис. 1.2, *б*);
- дискретні у множині та безперервні у часі (рис. 1.2, *в*);
- дискретні у множині та у часі, звичайні дискретні (рис. 1.2, *г*).

При математичному описі повідомлень формування дискретних повідомлень розглядають як послідовний випадковий вибір того або іншого символу з алфавіту джерела повідомлень, тобто формування дискретної випадкової послідовності. Формуванням безперервних повідомлень є вибір реалізацій (випадкових функцій) безперервного випадкового процесу. Але це питання є окремою темою і розглядається в теорії інформації.

Процес перетворення дискретних повідомлень в сигнал називається у широкому розумінні кодуванням. Проте у вузькому розумінні кодування – це відображення повідомлення у вигляді поєднання символів.

Розглянемо узагальнену структурну схему інформаційної системи, яка наведена на рис. 1.3.

Як показано на рис. 1.3, система обміну даними складається:

- з джерела інформації (передавач або лінійний формувач), яке перетворює повідомлення у форму, зручну для передачі по каналу зв'язку;
- з приймача, який приймає сигнали та перетворює їх зворотно у початкове повідомлення;

- з каналу зв'язку, по якому передаються сигнали. Це можуть бути провідні або безпровідні лінії зв'язку, а саме:
  - мідний дріт;
  - оптоволокно;
  - радіозв'язок;
  - супутниковий зв'язок.

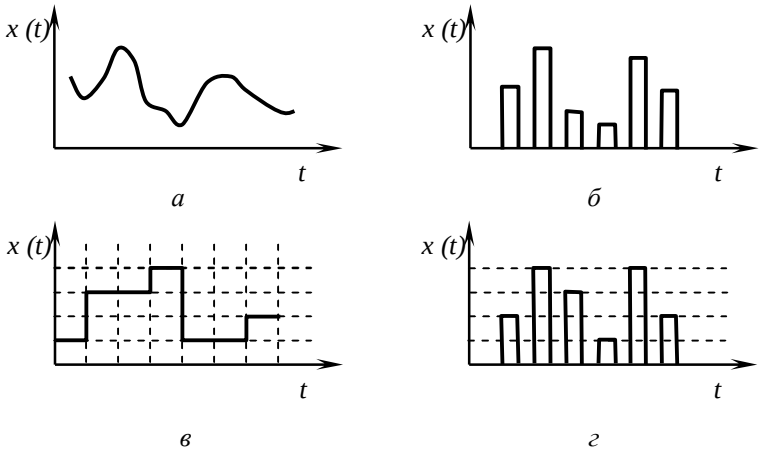


Рисунок 1.2 – Типи повідомлень в площині множин та у часі

На схемі (рис. 1.3) показано проходження повідомлення від джерела до одержувача з позначенням додаткових пристроїв, які здійснюють операції *кодування* та *модулювання*. Відзначимо, що в процесі передавання сигнал піддається ще іншим перетворенням, багато з яких є типовими, обов'язковими для різних систем електрозв'язку, незалежно від їх призначення та характеру повідомлень, які передаються.

Джерело повідомлення – це фізичний об'єкт, який формує конкретне повідомлення *x(t)* (люди, ПК, ПЛК, датчики). Приклади повідомлень – це різноманітний контент: мова, музика, фотографія, текст, рисунок або значення технологічного параметра. Перетворювачі повідомлення в електричний сигнал (мікрофон, датчик) перетворюють повідомлення *x(t)* на первинний сигнал *s(t)*. Наприклад, це перетворення значення тиску рідини всередині ємності в уніфікований електричний струмовий сигнал

4…20 мА або в цифровий код за допомогою аналого-цифрового перетворювача (АЦП). Сигнал – це матеріальний носій повідомлення. Модулятор здійснює перетворення первинного сигналу *s(t)* у вторинний сигнал *S(t)*, який більш зручний для передавання в середовищі поширення в умовах дії перешкод *n(t)*. Далі, на прийомному боці системи здійснюється зворотне перетворення.

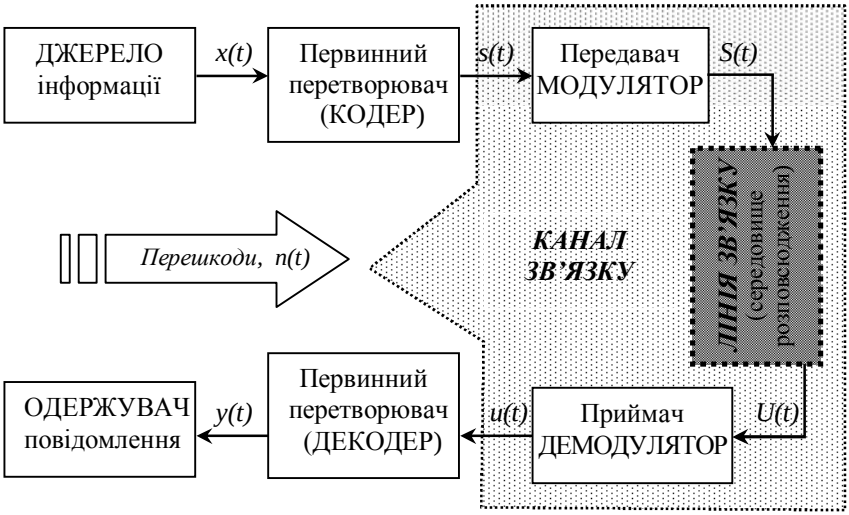


Рисунок 1.3 – Узагальнена структурна схема інформаційної системи

Кодер джерела служить для перетворення повідомлень в кодові символи з метою зменшення надмірності джерела повідомлення, тобто для забезпечення мінімуму середнього числа символів на одне повідомлення та його подання в зручній формі для подальших перетворень та передавання (наприклад, у вигляді двійкових чисел).

До передавача входить також *кодер каналу*, який призначений для введення надмірності з метою підвищення достовірності передавання даних, що дозволяє виявляти та виправляти помилки в канальному декодері. *Декодер каналу* забезпечує перевірку надмірного (перешкодостійкого) коду і перетворення його в послідовність первинного електричного сигналу коду без надмірності.

Декодер джерела – це пристрій для перетворення послідовності коду вже без надмірності в повідомлення.

У системах зв'язку прийнято розрізняти дві групи відносно самостійних пристроїв: *кодеки* та *модеми*. *Кодеком* називається сукупність кодера та декодера, які при двосторонньому зв'язку конструктивно об'єднані в один пристрій. *Модемом* називається конструктивно суміщена сукупність модулятора та демодулятора.

Розглянемо ще один компонент системи зв'язку, а саме канал зв'язку. Під *каналом зв'язку* розуміється сукупність засобів, які забезпечують передавання сигналу від деякого пункту *A* до пункту *B*. Пункти *A* і *B* можуть бути вибрані різним чином, залежно від завдання, що вирішується системою зв'язку. Залежно від виду вхідних та вихідних символів канал зв'язку може бути безперервним, дискретним або дискретно-безперервним. В одній і тій же схемі можна виділити як дискретний, так і безперервний канал, залежно від вибору точки розгляду системи.

Розглянемо детальніше деякі поняття та визначення, які пов'язані з інформаційними системами і які відомі з теорії інформації та кодування.

Інформація – це сукупність відомостей про які-небудь події, процеси, явища тощо, які розглядаються в аспекті їх передачі у просторі та у часі.

Повідомлення – це інформація, яка виражена в певній формі і призначена для передачі від джерела до одержувача.

Сигнал – це цілеспрямована зміна фізичного середовища або процесу, яке пов'язує джерело з одержувачем.

Отже, інформацію передають у вигляді повідомлень, які, в свою чергу, подаються сигналами, що є носіями інформації. Прикладами повідомлень можуть бути тексти телеграм, мова, музика, телевізійне зображення, дані на виході комп'ютера, параметри та команди в системі автоматичного управління технологічними об'єктами тощо. Основним видом сигналів в таких системах є електричні або оптичні сигнали. При цьому такі поняття, як кількість, достовірність і швидкість передачі інформації є основними при виборі параметрів протоколу та інтерфейсу.

Окрім названих термінів, в теорії інформації дається поняття про деякі інші терміни: *шум*, *перешкода*, *лінія* та *канал зв'язку*.

Шум – сторонні інформаційні потоки, які виникають в системі не у зв'язку з її цільовим призначенням. Зміни фізичного середовища, що викликаються шумом, пов'язують джерело з одержувачем, називаються *перешкодами*. Перешкоди бувають адитивні, мультиплікативні та комбіновані.

Канал зв'язку – сукупність технічних пристроїв та природного фізичного середовища, яка використовується для передачі і перетворення сигналів єдиної фізичної природи.

Лінія зв'язку – складова частина каналу зв'язку, в якому сигнал предається від передавача до приймача та зазнає лише природних змін (середовище розповсюдження).

Зрозуміло, що передавач та приймач повинні розуміти один одного. Для цього має бути вироблена угода за рядом чинників:

- тип використовуваних сигналів;
- визначення логічної «одиниці» та логічного «нуля»;
- коди, які подають ці символи;
- спосіб забезпечення синхронізації між передавачем та приймачем;
- порядок управління потоком даних;
- спосіб виявлення та виправлення помилок, які виникають в процесі передавання та поширення сигналів.

Фізичні чинники зазвичай називають *стандартом інтерфейсу*, всі останні чинники відносяться до *протоколу обміну*. Фізичний спосіб передачі даних по каналу зв'язку залежить від використовуваного середовища. Наприклад, двійкові значення «0» і «1» можуть виражатися присутністю або відсутністю напруги на мідному дроті, двома звуковими частотами, що генеруються і декодуються модемом, як це відбувається в телефонній лінії зв'язку, або шляхом модуляції променю світла, як це відбувається при передачі сигналу по оптоволокну. Таким чином, *інтерфейс* – комплекс засобів уніфікованого сполучення між складовими частинами системи обробки даних, що включає апаратні засоби і комунікаційний протокол.

Інформація завжди передається блоками даних залежно від процедур обміну між об'єктами, які називають *протоколами передачі даних*.

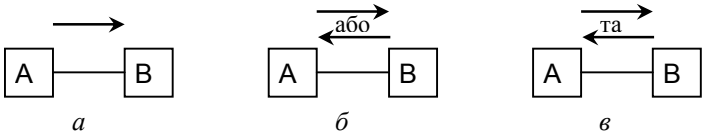
Отже, *протокол* – це сукупність правил, що встановлюють єдині принципи взаємодії підсистем та обмін даними між вузлами в мережі, тобто встановлюють формат та процедури обміну інформацією.

Введені терміни властиві певним технічним системам, наприклад телекомунікаційним. В подальшому інформаційні системи будуть розглядатись як складові компоненти комп'ютеризованих систем управління технологічними процесами.

1.2. Архітектура побудови та методи організації каналів зв'язку

Взагалі, через будь-який канал зв'язку дані можна передати трьома способами (рис. 1.4):

- симплексним (канал працює лише в одному напрямі);
- півдуплексним (канал по черзі працює в двох напрямах);
- дуплексним (канал одночасно доступний в двох напрямах).

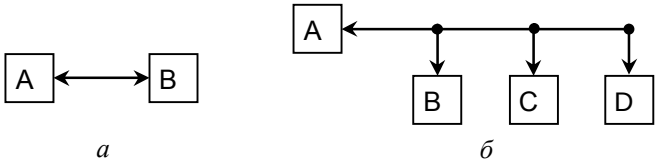


а *б* *в*

Рисунок 1.4 – Способи обміну даними:
а – симплекс; *б* – півдуплекс; *в* – дуплекс

Системи обміну даними класифікують за архітектурою побудови та способом з'єднання вузлів. Так, є системи для з'єднання лише двох вузлів («точка – точка») або з'єднання одного вузла з багатьма вузлами («точка – багато точок»). В першому випадку два пристрої, що взаємодіють, розділяють загальну лінію зв'язку (це спосіб обміну симплекс або півдуплекс). У другому випадку вважається, що дані передаються одним пристроєм, а приймаються декількома пристроями (можлива реалізація усіх трьох способів обміну даними).

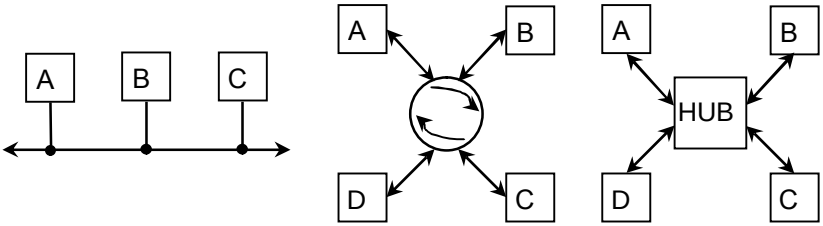
На рис. 1.5 наведено варіанти архітектури побудови мереж обміну даними. На базі цих двох архітектур будуються інші топології, які є їхньою варіацією. Взагалі є такі види топології побудови мереж – *шина*, *кільце*, *зірка* тощо.



а *б*

Рисунок 1.5 – Способи обміну даними:
а – «точка – точка»; *б* – «точка – багато точок»

На рис. 1.6 наведено види топології побудови мереж, яка визначає спосіб підключення та метод доступу вузла до лінії зв'язку. Але питання доступу до мережі буде розглянуто далі.



а *б* *в*

Рисунок 1.6 – Види топології побудови мереж:
а – шина; *б* – кільце; *в* – зірка

Зауважимо, що базові види топології мереж є основою побудови більш складних мереж, наприклад мереж типу *сота*, *дерево* тощо. При цьому не всі ці варіанти можуть бути реалізованими для послідовних інтерфейсів, наприклад, *зірка* властива лише технології *Ethernet*. Крім того, існує поділ видів топології на фізичну та логічну. *Фізична топологія* визначається власне фізичним з'єднанням вузлів у мережі. Проте *логічна топологія* визначається методом доступу до каналу зв'язку та напрямом передавання потоків даних між вузлами. Обидва види топології відносно незалежні одна від одної.

Так, в *логічній шині* інформація може передаватися від одного вузла усім іншим одночасно в даному сегменті мережі. При цьому *логічна шина* може бути побудована на *фізичній шині*, *зірці*, *дереві* або *соті*.

Логічне кільце передає дані послідовно від одного вузла наступному. Кожен вузол приймає кадри лише від попереднього і посилає лише подальшому вузлу по кільцю. Вузол транслює далі по мережі всі кадри, а обробляє ті, що адресуються лише йому. Реалізується таке кільце на *фізичному кільці* або на *зірці з кільцем усередині концентратора*. Сучасний підхід до побудови високопродуктивних мереж переносить велику частину функцій на центральні мережні пристрої (концентратори тощо). При цьому можна говорити про *логічну зірку*.

Насамкінець підведемо підсумки щодо варіантів побудови каналів зв'язку інформаційних мереж. На рис. 1.7 наведено загальну класифікацію раніше розглянутих каналів зв'язку.

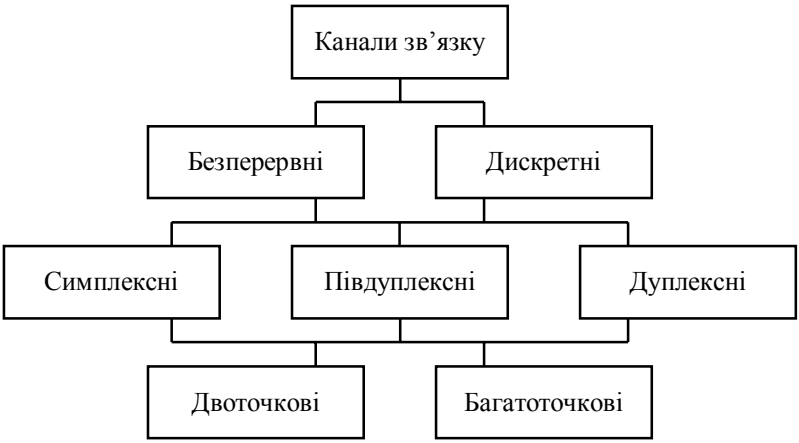


Рисунок 1.7 – Загальна класифікація каналів зв'язку

1.3. Загальна структура кадрів при передаванні інформації

Існує так званий стандартний базовий варіант структури інформаційного кадру будь-якого протоколу. Він складається з байтів синхронізації, адрес одержувача та відправника, байта даних та байта ознаки закінчення кадру та/або байта виявлення помилок. Також у процесі сеансу зв'язку обмін даними може бути синхронним або асинхронним відносно фіксації моменту передавання кадру. При асинхронному обміні момент

початку кадру визначається стартовим бітом. Синхронний обмін забезпечується передаванням в загальному потоці бітів спеціальних тактових імпульсів для синхронізації приймача з передавачем. Розглянемо структуру та склад інформаційних кадрів.

Асинхронний обмін здійснюється кадрами (які складаються з бітів або байтів). Він є більш поширеним і не вимагає додаткового обладнання для реалізації. Приймач не починає виявлення та оброблення сигналу до тих пір, поки він не прийме стартовий біт, який поданий переходом від негативного рівня (пауза) до протилежного (потенціал). Приймач обробляє дані по мірі їх надходження у часі за допомогою тактів свого генератора. Частота тактів однакова в приймачі та в передавачі. Вона визначає час передавання одного біта, а це безпосереднью пов'язано зі швидкістю передавання даних. Проте при збільшенні швидкості передачі (понад 100 кбіт/с) можливий збій і втрата даних. Взагалі говорять про швидкість символьну або бітову. В першому випадку мається на увазі інформаційна швидкість, у другому – це технічна швидкість, яка вимірюється у біт/с або бодах.

Формат асинхронного кадру показано на рис. 1.8 і має такий склад:

- стартовий біт (сигнали на початку кадру, визначають тактову частоту);
- дані (5...8 бітів);
- біт парності (біт виявлення помилки);
- стопові біти (1, 1.5 або 2 біти). Параметр «1.5» означає, що рівень утримується впродовж часу, який в 1,5 більше часу передавання одного біта.

Рисунок 1.8 – Формат кадру при асинхронному обміні даними

На рис. 1.8 зображені інформаційні та службові біти, причому задіяний біт паритету, який відповідає перевірці на непарність. В даному випадку значення біта паритету дорівнює «0» і передається позитивним рівнем сигналу, оскільки, в кадрі три біти мають статус «1», тобто в кадрі непарна кількість одиниць.

Щоб з асинхронного кадру можна було коректно витягувати дані, передавач та приймач мають бути однаково налагоджені. Це стосується швидкості передавання та складу кадру. Кожен символ має свій власний кадр, причому реальна швидкість передачі даних виходить менше швидкості передачі бітів в секунду. Наприклад, з урахуванням стартового біта, семи бітів даних, одного біта парності та одного стопового біта для передачі семи бітів даних всього необхідно передавати десять бітів. Таким чином, передача корисних даних складає 70 % від всієї швидкості передачі бітів.

При *синхронному* обміні синхронізація приймача здійснюється по тактових імпульсах передавача в структурі кадру. Це забезпечує передачу великих кадрів на довгі відстані. Біти упаковуються в пакети і передаються безперервно. Цей режим є більш складним ніж асинхронний.

Формат синхронного кадру складається з таких полів:

- преамбула (синхронізуючі біти);
- позначка початку роздільника даних (*SFD*);
- преамбула;
- адреса одержувача;
- адреса джерела;
- довжина даних;
- власне дані;
- контрольна послідовність для виявлення помилок (FCS).

Основні інформаційні параметри, які характеризують системи обміну інформацією, це:

- кількість інформації в повідомленнях;
- надмірність повідомлень;
- ентропія;
- продуктивність джерела повідомлень;
- швидкість передачі сигналів;

- швидкість передачі інформації;
- пропускна здатність каналу зв'язку.

Перші чотири характеристики належать до фундаментальних положень теорії інформації і не описують технічні параметри засобів зв'язку. Тому їх розгляд віднесемо на самостійне вивчення [1].

Розглянемо інформаційні характеристики для випадку дискретних повідомлень, оскільки саме дискретні повідомлення характерні для АСУ ТП і В.

1. *Швидкість передачі сигналів*. Величина, відповідна кількості змін фізичного сигналу в секунду (бод). Швидкість 1000 бодів відповідає імпульсам, які змінюються через 1 мс.

2. *Швидкість передачі інформації*. Величина виражається в біт/с або кратних одиницях: кбіт/с, Мбіт/с.

Для синхронних систем швидкість передачі бітів частіше перевищує швидкість передачі сигналу. Але для всіх систем швидкість передачі інформації менша, ніж швидкість передачі бітів. Існують способи передачі більше одного біта однією зміною сигналу. Проте швидкість передачі однозначно обмежена шириною смуги перепускання каналу зв'язку.

3. *Ширина смуги перепускання* обумовлена максимальним значенням частоти зміни сигналу в межах допустимого ослаблення. Теоретична межа максимальної швидкості визначається формулою Найквіста, яка залежіть від ширини смуги та кількості рівнів квантування при кодуванні повідомлення [1].

4. *Відношення сигнал-шум* (*S/N*) також обмежує швидкість передачі інформації.

5. *Перепускна здатність каналу* – це значення ефективної швидкості передачі, яка залежить від відношення кількості неінформативних параметрів до інформативних.

6. *Частота появи помилок* (*BER*) характеризується вірогідністю виникнення помилки на один біт інформації. Для промислових мереж дуже важлива відсутність помилок при управлінні ТП. Тому їх розробляють з урахуванням максимальної можливої надійності, що зумовило невелику інформаційну швидкість.

1.4. Методи кодування повідомлень джерела

Застосування кодування повідомлень джерела дозволяє їх перетворити в формалізовані посилання за допомогою так званого *алфавіту коду*. Важливо, що кількість символів цього алфавіту значно менша ніж, наприклад кількість звичайного мовного алфавіту.

Розглянемо деякі коди та їхні характеристики.

1. *Коди Морзе та Бодо.* Ці коди є найстарішими та винайдені відомими інженерами-винахідниками у галузі зв'язку Морзе та Бодо. Коди мають майже однаковий алфавіт – символи «точка», «тире» та «пропуск». Проте код Морзе, на відміну від коду Бодо, є нерівномірним. Нерівномірність пов'язана з різною кількістю символів коду в одному повідомленні відносно кількості символів первинного алфавіту. За допомогою п'яти бітів в коді Бодо можна подати 32 (25) символів. Він підходить лише для передачі букв, цифр та знаків пунктуації (наприклад, це телетайп). Сучасна версія коду адаптована організацією *ITU* (англ. *International Telecommunication Union* – міжнародна організація, яка визначає рекомендації у галузі телекомунікації та радіо) в якості стандарту із застосуванням символу зміни регістра.

2. Код *ASCII* (*American Standard Code for Information Interchange* – американський стандартний код для обміну інформацією) є класичним найбільш поширеним набором символів, який використовують в різних галузях телекомунікацій. Також цей код поширений в комп'ютеризованих системах управління технологічними процесами.

Код використовує рядок з 7 бітів, що дає можливість кодування 128 символів, в які входять комбінації 0000000…1111111:

- верхній та нижній регістри букв англійського алфавіту;
- цифри 0…9;
- знаки пунктуації та спеціальні символи;
- службові символи (усього 32 символи), які не зображуються, але виконують функції управляючих та службових символів.

У табл. 1.1 наведена скорочена версія *ASCII*-коду, в якій усі символи та їхні кодові комбінації подані у *HEX*-форматі та двійковому форматі у вигляді матриці. Дана таблиця побудована таким чином, що на перетині

стовпчиків та рядків показано символ, який кодується відповідно у *HEX*-форматі та двійковому форматі. Причому, стовпчики у верхньому рядку нумеровані за порядком зліва направо і відповідають трьом старшим бітам коду, де саме старший біт знаходиться зліва (сьомий за порядком) позначений як MSB. Рядки нумеровані в лівому стовпчику теж за порядком, від 0 до 15 в десятковому значенні, тому використано чотири молодших біти, молодший позначений як LSB.

Таблиця 1.1 – Таблиця відповідності *ASCII*-кодів символам

		DEC/ HEX	0	1	2	3	4	5	6	MSB 7
DEC	HEX	BIN	000	001	010	011	100	101	110	111
0	0	0000	NUL	DEL	space	0	@	P	`	p
1	1	0001	SOH	DC1	!	1	A	Q	a	q
2	2	0010	STX	DC2	“	2	B	R	b	r
3	3	0011	ETX	DC3	#	3	C	S	c	s
4	4	0100	EOT	DC4	$	4	D	T	d	t
5	5	0101	ENQ	NAK	%	5	E	U	e	u
6	6	0110	ACK	SYN	&	6	F	V	f	v
7	7	0111	BEL	ETB	`	7	G	W	g	w
8	8	1000	BS	CAN	(	8	H	X	H	x
9	9	1001	HT	EM	)	9	I	Y	I	y
10	A	1010	LF	SUB	*	:	J	Z	j	z
11	B	1011	VT	ESC	+	;	K	[	k	{
12	C	1100	FF	FF	,	<	L	\	l	|
13	D	1101	CR	GS	-	=	M	]	m	}
14	E	1110	SO	RS	.	>	N	^	n	~
15 LSB	F	1111	SI	US	/	?	O	–	o	DEL

Після перетворення повідомлення джерела канал зв’язку обробляє вже 16-ричні значення (00…7F). Для адаптування коду до формату даних в комп’ютері, де частіше використовують байт як одиницю оброблення, до коду додають восьмий (старший) біт. При зворотному перетворенні цей біт ігнорується.

На цей час існує низка кодів, які є модифікацією коду *ASCII*. Це насамперед так звані кодові таблиці для операційних систем, які доповню-

ють англійський алфавіт символами національного алфавіту, наприклад, українського. Є також кодові таблиці, які мають такі ж самі кодові комбінації що і код *ASCII*, але відповідають іншим символам. Це стандарти *ANSY-X3.4, ISO-646, ITU № 5*.

На рис. 1.9 показано склад кадру, який за допомогою таблиці *ASCII*-коду є перетворенням англійського слова «DATA». Зауважимо, що є деякі особливості використання *ASCII*-коду та передавання повідомлень по каналу зв'язку. По-перше, символи повідомлення відправляються в тому порядку, в якому вони сформовані джерелом, тобто в слові «DATA» порядок відправлення символів такий: спочатку літера «D», далі – інші. По-друге, 7-розрядний двійковий код передається в зворотному порядку, тобто спочатку молодший біт (LSB), далі передаються проміжні, закінчує послідовність старший біт (MSB). Так, символу «T» в *ASCII*-коді відповідає така послідовність в *HEX*-форматі «54» і в BIN-форматі «1010100». Проте в канал буде відправлена така послідовність – «0010101». В кадрі присутні три одиниці та здійснюється перевірка на непарність, тому відповідній біт парності має статус «0».

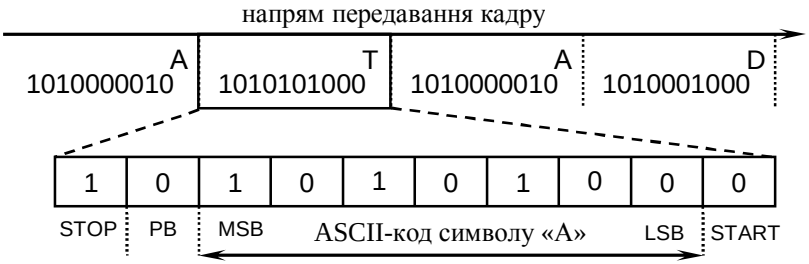


Рисунок 1.9 – Формат кадру при передаванні *ASCII*-символу

3. *Код Грея*. Іноді бінарні коди не відповідають технічним вимогам до апаратних засобів систем автоматизації, оскільки при послідовному збільшенні на одиницю не завжди змінюється лише один розряд (біт). Наприклад, процес інкрементування двійкового числа «0011» приведе до отримання числа «0100», тобто відбудеться зміна значення в трьох розрядах. Так, при використанні датчика кутового положення вала для вимірювання кутової швидкості використовують імпульсний метод. Тобто

обертання вала змушує датчик генерувати послідовність імпульсів, які, в свою чергу, рахуються. Процес підрахунку імпульсів є інкрементальним додаванням. Тому потрібно щоб додавання чергового імпульсу змінювало статус лише одного розряду. Зазвичай використовують такі датчики з фіксованою кількістю імпульсів на один оборот вала. В табл. 1.2 наведено чотирирозрядний код Грея.

Таблиця 1.2 – Таблиця відповідності коду Грея десятковим числам

Число в *DEC*-форматі	Кодова комбінація Грея
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

3. *Двійково-десятковий код (BCD).* Цей код розширює можливості звичайного бінарного коду. Якщо кожному окремому розряду десяткового числа від 0 до 9 поставити у відповідність чотирирозрядну двійкову комбінацію, то такими тетрадами можливе передавання інформації на 7-сегментні індикатори. В такому коді заборонені комбінації двійкового коду, які відповідають числам від 10 до 15, що в *HEX*-форматі має значення від «A» до «F».

1.5. Методи кодування повідомлень каналу

Далі коротко розглянемо методи подання та кодування повідо-млень каналу в системах обміну інформацією. На фізичному рівні кожному біту інформації ставиться у відповідність деякий електричний (оптичний, інфрачервоний) сигнал. Таке перетворення називають *модуляцією*. Існує два основних види модуляції: *аналогова* та *цифрова*. При аналоговій модуляції фізичний сигнал несе інформацію про початок та кінець кожного наступного біта, при цифровому кодуванні ця умова не обов'язково виконується. Відповідно виділяють:

- коди, які самі синхронізуються;
- коди, які самі не синхронізуються.

Також відзначимо, що існує так зване логічне кодування даних (перекодування даних), яке потрібне для первинного поліпшення характеристик передавання та створення умов для нормальної роботи апаратури передавання даних. Цей вид кодування може реалізовуватись або за рахунок використання так званих надмірних кодів, або за допомогою – скремблювання.

Логічне кодування має такі переваги:

1. Усунення довгих послідовностей нулів та одиниць – після такого логічного кодування можна використовувати коди, які не можуть самостійно синхронізуватись для подальшого передавання по каналу зв'язку.

2. Надання приймачу можливості розпізнавати і, можливо, усувати помилки в послідовності бітів.

Коди з надмірністю формуються шляхом заміни початкової послідовності за допомогою таблиці перекодування після її розбиття на блоки однакової довжини. Наприклад, код *4B/5B* замінює кожні чотири біти на п'ять бітів і далі передає послідовність в канал. Цей код застосовується в мережах *FDDI* та *Fast Ethernet*.

У процесі скремблювання розраховується функція, яка залежить від поточного значення біта з початкової послідовності та вже переданих значень.

Типовий канал обміну даними *на фізичному рівні* визначає *електричні та механічні параметри інтерфейсу* між термінальним устаткуван-

ням *(DTE – Data Terminal Equipment)* та апаратурою передачі даних *(DCE – Data Circuit Equipment).* Апаратура передавання даних (модеми) потрібна для підключення термінального обладнання (ПК, ПЛК тощо) до ліній зв'язку (наприклад, телефонних ліній фіксованого зв'язку).

Як зазначалося раніше, дуже важливим є принцип подання семантичної інформації, яку генерує джерело. На даний час існує декілька варіантів кодування (подання) інформації. Усі вони пов'язані з технічною реалізацією передавача та приймача, тобто пристрою *DCE*. Так, існують пристрої для широкосмугового та вузькосмугового передавання аналогових і цифрових сигналів.

Умовно варіанти передавання та кодування інформації можна уявити так, як це наведено в табл. 1.3. Надамо деякі пояснення щодо змісту табл. 1.3. Для передавання даних широкосмуговим сигналом залежно від типу повідомлення використовують перетворювачі для аналогових (перший квадрант) та модеми – для цифрових даних (другий квадрант). В даному випадку для переносу інформації по лінії зв'язку використовують електромагнітні хвилі з параметрами, які постійно змінюються. Це може бути синусоподібний сигнал в якості несучого, який модулюється інформаційними символами. Прикладом такої реалізації є передавання інформації в аналоговій формі через широкосмуговий канал. Це мова, яка передається по телефонній лінії за такою схемою: звуковий сигнал (*DTE*) → мікрофон (*DCE*) → аналоговий сигнал (перший квадрант). Передавання цифрової інформації (наприклад, параметри технологічного процесу) широкосмуговим каналом здійснюється за такою схемою: виміри модулюють сигнал-носій в передавачі, а потім прийнятий сигнал демодулюється в приймачі. Ці функції з обох боків лінії зв'язку здійснює пристрій під назвою *модем* (другий квадрант). Схема каналу зв'язку така: ПЛК (*DTE*) → модем (*DCE*) → аналоговий канал (звичайна телефонна лінія фіксованого зв'язку).

Таблиця 1.3 – Способи передавання та кодування інформації

	Тип передавача	*Широкосмугове передавання*	*Вузькосмугове передавання*
Тип повідомлення	*Аналоговий*	I – Перетворювач	III – Кодек
	Цифровий	II – Модем	IV – *CSU/DSU*

У третьому та четвертому квадрантах показані варіанти передавання інформації за допомогою вузькосмугових носіїв. В даному випадку дискретні сигнали подаються уніполярним або біполярним станом каналу зв'язку. Так, в третьому квадранті аналогова інформація передається по цифровому каналу за допомогою так званого *кодека* за такою схемою: відеоінформація (*DTE*) → кодек (*DCE*) → цифрова лінія (*T1*). Передавання цифрової інформації через цифровий канал здійснюється за допомогою спеціальних пристроїв, які є аналогами пристроїв *DTE* та *DCE*. Це пристрої *CSU* (*Channel Service Unit*) та *DSU* (*Data Service Unit*), в яких інформація перетворюється з формату пристрою-джерела в формат каналу зв'язку.

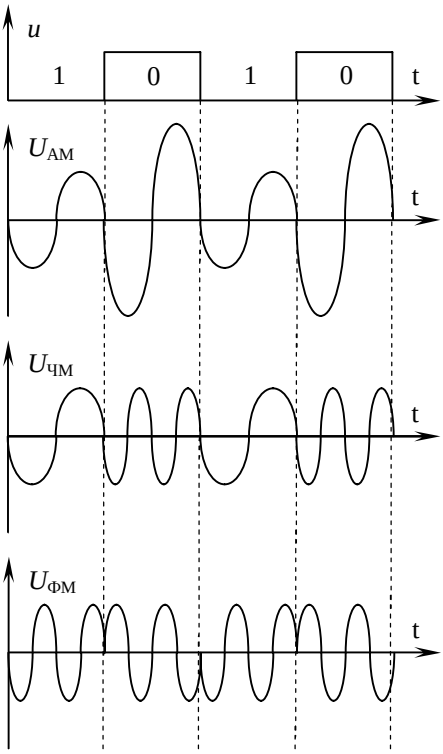


Рисунок 1.10 – Види аналогової модуляції

У випадку аналогової модуляції корисна інформація модулює безперервний гармонійний синусний сигнал шляхом впливу на його амплітуду, частоту або фазу. При цьому інші параметри сигналу залишаються постійними. Це відповідно амплітудна (*АМ*), частотна (*ЧМ*) або фазова (*ФМ*) модуляція. На рис. 1.10 показані діаграми з відображенням вказаних видів аналогової модуляції.

Отже, розглянемо названі види аналогової модуляції.

1. *Амплітудна модуляція* (АМ) – для передавання «одиниці» обирається один рівень амплітуди, для «нуля» – інший. При цьому інші параметри синусоїдального сигналу (частота та фаза) залишаються постійними. У чистому вигляді АМ рідко використовується через низьку перешкодостійкість.

2. *Частотна модуляція* (ЧМ) – «одиниця» передається сигналом однієї частоти, «нуль» – сигналом іншої частоти. Тут відповідно амплітуда та фаза сигналу постійні. Ця модуляція використовується в модемах з низькою швидкістю.

3. *Фазова модуляція* (ФМ) – здійснюється шляхом змінення фази сигналу на протилежний або для «одиниці» та «нуля» використовують різні початкові фази сигналу. Така модуляція теж використовується в модемах з низькою швидкістю передавання інформації.

При цифровому кодуванні використовують послідовність прямокутних імпульсів. Відзначимо, що цей спосіб передавання інформації рідко використовують в системах обміну даними, які входять до структури АСУ ТП і В.

Розглянемо основні види цифрового кодування.

1. *Not Return to Zero* (*NRZ*) – код без повернення до нульового рівня. В даному випадку нульовому біту відповідає високий рівень напруги в кабелі, одиничному – низький рівень (або навпаки). Протягом бітового інтервалу (часу передавання одного біта) змін рівня сигналу не відбувається. Це код, який не може синхронізуватись самостійно. В такому випадку неможливо визначити початок та кінець даних, тобто відокремити окремі кадри. Приклад застосування такого коду – це асинхронний послідовний інтерфейс *RS-232*. На рис. 1.10 показано передавання інформації кодом без повернення до нуля.

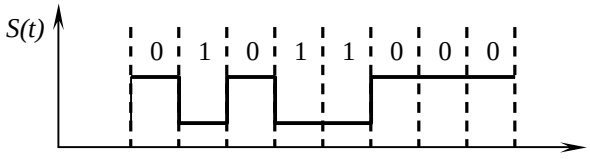


Рисунок 1.11 – Код без повернення до нуля

2. *Not Return to Zero with ones Inverted* (*NRZi*) – код без повернення до нульового рівня з інверсією при одиниці. Надходження одиничного біта перемикає рівень напруги сигналу на початку бітового інтервалу, а нульового – залишає рівень без зміни. Це теж код, який не може самостійно синхронізуватися і відповідно потребує додаткових засобів для

розділення кадрів. Даний вид кодування використаний в послідовному інтерфейсі *USB*. На рис. 1.12 показаний принцип кодування без повернення до нуля з інверсією по одиниці.

Рисунок 1.12 – Код без повернення до нуля з інверсією

3. *Multi-Level Transition-3* (*MLT-3*) – це багаторівневе кодування даних, багаторівнева модуляція. Нульовому біту відповідає збереження рівня напруги. Одиничному біту на початку бітового інтервалу відповідає перемикання рівня напруги на наступний за схемою: +U, 0, -U, 0, +U, 0. Цей код потребує меншої смуги перепускання ніж у коді *NRZ*, але він теж не може синхронізуватись і неможливо визначити межі кадрів. На рис. 1.13 показаний принцип багаторівневого кодування.

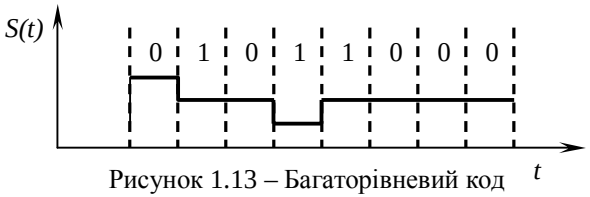


Рисунок 1.13 – Багаторівневий код

4. *Return to Zero* (*RZ*) – код з поверненням до нуля. Нульовому біту відповідає позитивне перемикання рівня напруги на початку бітового інтервалу, одиничному – негативне перемикання. В середині бітового інтервалу відбувається повернення до початкового рівня сигналу. Код може самостійно синхронізуватися. Приймач може визначити початок та кінець кадру з даними. На рис. 1.14 показано діаграму коду з поверненням до нуля.

5. *«Манчестерський» код* (*Манчестер-II*). Цей вид кодування був спочатку запропонований студентами технологічного університету м. Манчестер (Англія), а потім доопрацьований для подальшого використ-

тання. В цьому коді нульовому біту відповідає позитивне перемикання в центрі бітового інтервалу, одиничному – негативне перемикання. Використовується лише два рівні сигналу. Цей код, на відміну від попередніх, може самостійно синхронізуватися. В такому випадку приймач може визначити початок і кінець кадру з даними. Приклади застосування – це технології *Ethernet* та *Token Ring*. На рис. 1.15 показана діаграма «манчестерського» коду.

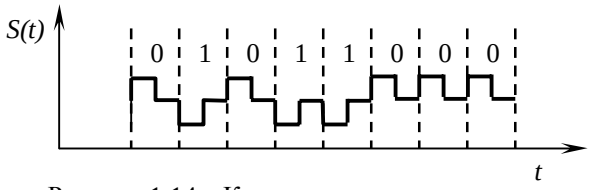


Рисунок 1.14 – Код з поверненням до нуля

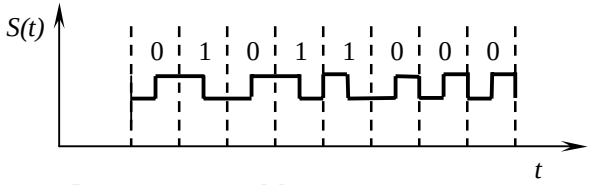


Рисунок 1.15 – «Манчестерський» код

6. *Код 2B1Q* належить до складних кодів, які за допомогою використання чотирьох рівнів напруги кодують два біти даних за такою схемою: 00 – *-U1*, 01 – *-U2*, 10 – *+U2*, 11 – *+U1*. Цей код вимагає від передавача збільшеної потужності, щоб приймач мав змогу чітко розрізняти усі рівні. Цей код не може синхронізуватись і тому неможливо визначити межі кадрів. На рис. 1.16 показана діаграма складного коду типу *2B1Q*.

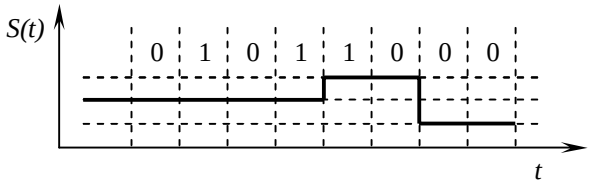


Рисунок 1.16 – Код *2B1Q*

1.6. Методи доступу до каналів зв'язку

Метод доступу до каналів зв'язку – це спосіб визначення того, яка з робочих станцій (вузлів) має використовувати лінію зв'язку та власне порядок управління процедурою доступу до фізичного каналу. Метод доступу та протокол обміну є взаємопов'язаними поняттями. Методи доступу поділяють на детерміновані та імовірнісні. Детермінованість методу полягає у централізованому доступі до каналу зв'язку, тобто процес обміну даними керується з боку головного пристрою. Імовірнісний спосіб характерний для мереж, в яких рішення про обмін кожен вузол приймає незалежно від його статусу в мережі.

Залежно від способу з'єднання вузлів можна виділити дві технології передачі:

- широкомовні мережі або мережі із загальним доступом;
- мережі з передачею від вузла до вузла.

Широкомовні мережі користуються спільним каналом зв'язку, який сумісно використовується усіма вузлами в мережі. Повідомлення, які відправляються одним вузлом, отримують всі інші вузли, а для ідентифікації одержувача використовують спеціальне поле адреси в пові-домленні або спеціальний символ. Якщо повідомлення надсилається усім вузлам, то така передача називається *широкомовною*.

Мережі з передачею від вузла до вузла складаються з великої кількості з'єднаних пар вузлів. Повідомлення під час передавання проходить послідовно через декілька пар вузлів.

Для широкомовних мереж (шинна та деревоподібна топологія) необхідно визначити порядок доступу до єдиного каналу. Широкомовні мережі, які використовують як носій сигналу металевий провідник, мають шинну топологію. Якщо у мережі з шинною топологією не буде чітко визначено, хто в конкретний момент може займати шину для передачі, то може виникнути момент одночасної передачі бітової послідоності двома або кількома передавачами. Така ситуація називається колізією (*collision*) або конфліктом. У цьому випадку приймачі не зможуть визначити, який біт був переданий, оскільки рівень сигналу буде результатом

дій декількох передавачів. Існує декілька варіантів вирішення порядку доступу до шини:

- централізований «ведучий – ведений»;
- централізований метод з арбітражем;
- децентралізований з передаванням маркера;
- множинний з випадковим доступом (*CSMA*);
- множинний з використанням поля арбітражу;
- метод з часовим розділенням (*TDMA*);
- метод з частотним розділенням каналу (*FDMA*).

Існують також гібридні методи, в яких поєднані наведені вище методи доступу.

Найбільш поширеним методом, який використовують в послідовних каналах зв'язку, є мережі з централізованим методом доступу – мережі типу «ведучий – ведений» («*Master – Slave*», «головний – підпорядкований), у яких право на управління володінням шиною передається «ведучому» вузлу, а «ведені» займають шину лише з його дозволу. Останні мають унікальну адресу (адреса «веденого»), за допомогою якої «ведучий» ідентифікує кому надсилається повідомлення. Прикладом використання даного методу є протоколи *ModBus*, *OWEN* та *DCON*, які будуть розглянуті у третьому розділі посібника.

1.7. Багаторівнева модель зв'язку

Для спрощення структури більшість мереж організується в набір *рівнів* або *шарів* (*layers*), кожен з яких з'єднується з сусіднім (верхнім або нижнім). Кількість рівнів, їх назва та склад залежать від побудови мережі. Однак у всіх мережах метою кожного рівня є надання деяких сервісів для вищого рівня та підтримання зв'язку з таким же рівнем на іншому пристрої. Правила і домовленості, які використовують в цьому спілкуванні, називаються *протоколами рівня* (*layer protocol*). Але дані не пересилаються безпосередньо між однаковими рівнями. Передача між вузлами відбувається таким чином (див. рис. 1.17): дані, які необхідно передати, надходять до самого верхнього рівня, який обробляє їх і передає нижчому рівню до 1-го рівня, який займається безпосередньо передачею да-

них по *фізичному середовищу*. На іншому вузлі дані, які приймаються, обробляються кожним рівнем до верхнього. Оброблення даних на кожному рівні здійснюється відповідно до протоколу. Між кожною парою суміжних рівнів знаходиться *інтерфейс* (*interface*), який визначає набір примітивних операцій та функцій, які надаються нижнім рівнем верхньому. Набір рівнів та протоколів називається *архітектурою мережі*. Список протоколів, які використовуються системою, по одному на рівень, називається *стеком протоколів*.

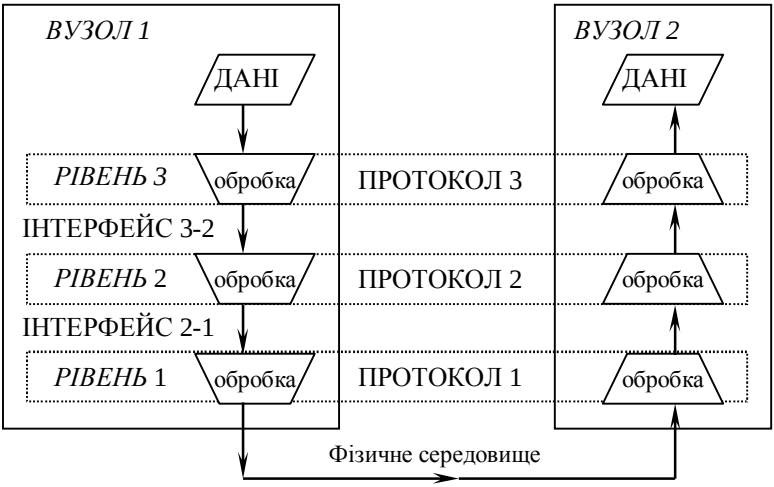


Рисунок 1.17 – Побудова мережі з багаторівневою архітектурою

Оброблення даних на кожному з рівнів включає перетворення інформації з одного виду в інший, розбивання даних на частини та додавання службової інформації.

При розробці мереж на кожному з рівнів визначатимуть:

- механізм ідентифікації відправників та отримувачів, систему адресації вузлів;
- правила передавання даних: напрямок, логічні канали;
- контроль за помилками: методи визначення та дії при їхньому виникненні;

- правила розбиття даних на їх об'єднання;
- управління потоком даних, узгодження параметрів передавання;
- маршрутизація, тобто визначення маршруту просування даних по мережі.

Як було зазначено вище, кожен з рівнів в мережі надає набір сервісів (служб) для реалізації певних завдань. *Сервіс* (*Service*) – це набір операцій, які нижній рівень надає верхньому. Сервіс визначає, які саме операції рівень буде виконувати, але не робить застереження яким чином.

Коротко ознайомимося з типами сервісів. За видом з'єднання сервіси можна поділити на два типи:

- *орієнтовані на з'єднання*, які передбачають попереднє створення каналу, а потім передачу по ньому даних;
- *без установки з'єднання*, які передбачають механізм доставки даних без попереднього створення каналу та попередньої домовленості.

За якістю обслуговування сервіси діляться на *надійні* та *ненадійні*. *Надійні сервіси* передбачають контроль за доставку даних отримувачу за допомогою відповідного підтвердження, якого не потребують *ненадійні сервіси*. У випадку останніх дані, які не дійшли до отримувача, повторно вислані не будуть.

Надійні сервіси, орієнтовані на з'єднання, бувають двох типів: послідовності повідомлень та байтові потоки. В першому варіанті між повідомленнями зберігаються межі, в другому – набір повідомлень являє собою неперервний потік байтів.

Ненадійні сервіси, без установки з'єднання, називаються сервісами *дейтаграм* або *дейтаграмними сервісами* (*datagram service*). Надійний сервіс без установки з'єднання називається сервісом дейтаграм з підтвердженням.

Існують типи сервісів, які називаються сервісами «запит – відповідь» («*request – response*»), в якому відправник відправляє дейтаграми, які вміщують запити, а одержувач на ці запити відповідає. Такі сервіси часто використовують для реалізації моделі обміну «клієнт – сервер». Типи сервісів та з'єднань зведені в табл. 1.4.

Таблиця 1.4 – Типи сервісів та з'єднань

Тип з'єднання	Сервіс
Орієнтоване на з'єднання	Надійна послідовність повідомлень
	Надійний потік байтів
	Ненадійне передавання бітів
Без встановлення з'єднання	Ненадійна дейтаграма
	Дейтаграма з підтвердженням
	Запит – Відповідь

1.8. Еталонна модель взаємодії відкритих систем

Об'єднання в одну мережу декількох пристроїв – це лише початковий крок до організації їх ефективної і надійної взаємодії. Там, де комунікаційні системи гомогенні (однорідні), тобто побудовані з пристроїв одного виробника, там апаратно-програмні проблеми, як правило, вирішені. Але коли мова йде про побудову систем з використанням пристроїв різних виробників (гетерогенних), то неминуче постають питання їх сумісності.

Унікальні системи, тобто системи, що працюють за унікальними протоколами зв'язку та вироблені однією компанією, отримали назву «*закритих систем*». Стосовно промислових мереж, частинні розв'язки – це інтелектуальна власність окремих компаній, і використання таких технологій обмежується необхідністю отримання ліцензійного права користування. Більшість таких систем виникли, коли проблеми інтеграції виробів від різних виробників не вважалися актуальними.

Альтернативою може бути використання принципу «*відкритих систем*». Мережа вважається «*відкритою*», якщо вона задовольняє таким критеріям:

- наявність повних опублікованих специфікацій з можливістю їх придбання за розумні гроші;
- наявність критичного мінімуму доступних компонентів від ряду незалежних постачальників;

- організація певного процесу ратифікації можливих доповнень до стандартів і специфікацій.

Коротше це можна сформулювати так: кожен охочий має можливість використовувати те, що вже напрацьоване, або робити власні розробки, у тому числі і такі, які можуть використовуватися іншими.

Якщо деяка мережна технологія відноситься до «*відкритих систем*», то вона повинна володіти таким рядом принципових якостей:

- *interconnectivity*, тобто можливістю вільного фізичного включення в загальну мережу пристроїв від різних виробників;
- *interoperability*, тобто можливістю побудови працездатної мережі на основі включення компонентів від різних постачальників;
- *interchangeability*, тобто можливістю заміни компонентів аналогічними пристроями від інших виробників.

Для полегшення взаємодії між устаткуванням різних виробників і збільшення виграшу від зростання масштабів виробництва потрібні стандарти. Через складність завдання комунікації одного стандарту буде недостатньо. Тому функції повинні розділятися на безліч керованих частин і бути організовані у вигляді архітектури зв'язку, який потім створить основу для стандартизації.

Як спосіб структуризації був вибраний досить поширений метод розбиття на рівні. Цей метод полягає у такому: функції зв'язку розділяються на ієрархічний набір рівнів, і кожен рівень виконує співвіднесений з ним підклас функцій, потрібних для повідомлення з іншими системами. При цьому кожен рівень доручає нижньому рівню завдання виконання більш елементарних функцій і заховування деталей цих функцій. В той же час кожен рівень надає послуги вищому рівню. В ідеальній ситуації рівні можна визначити так, що зміни в одному рівні не спричинять змін в інших. Таким чином, ми розбиваємо одне завдання на деяку кількість менших завдань, які легше піддаються управлінню. Головним завданням було визначення послідовності рівнів і послуг, що надаються кожним з них. Розбиття повинне логічно групувати функції і містити таку кількість рівнів, щоб кожен з них був достатньо малий, щоб піддаватися управлінню.

При цьому кількість рівнів не повинна бути такою великою, щоб витрати на їх обробку були не дуже обтяжливими.

На сьогоднішній день найбільш популярні мережі побудовані за принципом відкритості і за основу використовують так звану еталонну модель взаємодії відкритих систем – *RM OSI* (*Open Systems Interconnection Basic Reference Model*). Модель була запропонована Міжнародною організацією стандартизації (*ISO*) в 1978 році. З того часу вона декілька разів переглядалася, востаннє у 1995 році. Дана модель закріплена стандартом організації *ISO* під назвою *ISO 7498* [3].

Модель створена з метою вирішення проблеми взаємодії відкритих систем з різними видами обчислювального та комунікаційного устаткування, які працюють за різними стандартами і протоколами. Лише при використанні принципу «відкритості систем» інтеграція виробів різних виробників в одну інформаційну мережу може бути вирішена без особливих проблем. Модель базується на стеку протоколів *OSI*, але на цей час ці протоколи продовжують удосконалювати, тому їхні функції мають рекомендований характер. Крім того, модель є лише каркасом і не вимагає обов'язкового виконання, тобто має на меті лише рекомендації. В цілому модель побудована на трьох концепціях, а саме – поняттях «сервіс», «протокол» та «інтерфейс». *Сервіс* визначає, що саме робить рівень (але, зауважимо, не те як це він робить) і яким чином доступитися до функцій цього сервісу. *Інтерфейс* рівня визначає спосіб доступу до рівня для розміщених вище процесів, тобто параметри та отриманий результат. *Протоколи* – це спосіб реалізації сервісів, який надає кожний рівень. Це значить, що якщо верхній рівень захотів скористатися одним із сервісів нижнього рівня, він звертається до нього через інтерфейс, а рівень вибирає саме яким протоколом із доступних на даному рівні йому скористатися. В моделі є лише один рівень для організації фізичного канала між вузлами, тобто рівень зв'язку з середовищем розповсюдження. Інші рівні моделі мають лише *віртуальний канал*. Усі проміжні рівні, крім верхнього та нижнього, мають по два інтерфейси.

Отже, модель взаємозв'язку – це абстрактна модель для мережних комунікацій і розроблення мережних протоколів, яка являє собою бага-

торівневий підхід до інформаційної мережі. Кожен рівень обслуговує свою частину процесу взаємодії. Завдяки такій структурі спільна робота мережного устаткування і програмного забезпечення стає набагато простішою та зрозумілішою.

Ієрархія моделі *OSI* показана в табл. 1.5. Модель згідно із стандартом описує сім рівнів, для яких в таблиці і в тексті описані їхні функції.

Таблиця 1.5 – Побудова еталонної моделі взаємодії відкритих систем

Рівні моделі		Типи даних	Функції рівня
Рівні, які не залежать від реалізації каналу зв'язку (*Host layers*)	7. Прикладний (*Application*)	Дані	Доступ до мережних сервісів
	6. Подання (*Presentation*)	Дані	Подання та кодування даних
	5. Сеансовий (*Session*)	Дані	Управління сеансом зв'язку
З'єднувальний рівень	4. Транспортний (*Transport*)	Сегменти	Забезпечення з'єднання двох хостів та надійність
Рівні, які залежать від реалізації каналу зв'язку (*Media layers*)	3. Мережний (*Network*)	Пакети/ Датаграми	Визначення маршруту та логічне адресування
	2. Канальний (*Data Link*)	Біти/ Кадри	Фізичне адресування та доступ до каналу зв'язку
	1. Фізичний (*Physical*)	Біти	Безпосередня передача та прийом бітів у середовищі

Виходячи з того, що модель слугує для виконання потреб кінцевих користувачів, розглядати її рівні ми почнемо з верхнього (сьомого) рівня, який відповідає за обмін даними між прикладними сутностями, тобто джерелом інформації та її одержувачем. Отже, детальніше рівні моделі характеризуються так.

7. Прикладний рівень (*Application Layer*) повинен забезпечити одній прикладній програмі доступ до об'єктів іншої прикладної програми через систему домовленостей. Для реалізації завдань прикладного рівня формуються запити та відповіді прикладних програм до рівня подання.

6. Рівень подання (*Presentation Layer*). Служить для перетворення форматів даних із одного в інший. На цьому рівні повідомлення кодуються та стискаються згідно з правилами синтаксису даних для подальшого використання на транспортному рівні.

5. Сеансовий рівень (*Session Layer*) дозволяє організовувати сеанси обміну між прикладними програми (встановлення та розрив з'єднання тощо).

Дозволяє установлювати сеанси прикладних програм між собою. Надає такі сервіси: управління діалогом (слідкування за послідовністю передачі даних), управління маркерами (запобігання виконання критичної операції декількома програмами) і синхронізація (використання службових міток в середині довгих повідомлень для відновлення передачі з обірваного через помилку повідомлення).

4. Транспортний рівень (*Transport Layer*) повинен забезпечити достовірну доставку даних від однієї програми (прикладного процесу) до іншої, які функціонують на одному вузлі або на різних вузлах в мережі.

Основна функція рівня – це приймання даних від сеансового рівня, при необхідності розбиття на частини, передавання їх мережному рівню і гарантування прибуття їх у правильному порядку до місця призначення. Цей рівень забезпечує передачу даних безпосередньо між прикладними програмами.

3. Мережний рівень *(Network Layer)* повинен забезпечити достовірну доставку даних від одного вузла до іншого в різних мережах, об'єднаних в одну інтермережу.

Основне завдання – це визначення маршруту пересилання пакета від джерела до приймача. Мережа може являти собою об'єднання підмереж, між якими виникає необхідність в обміні даними. Оскільки підмережі можуть бути різнорідними за природою – завдання мережного рівня також забезпечити прозорість між ними. Окрім цих функцій, на цей рівень покладаються також функції визначення часових затримок пакетів в мережах, питання синхронізації, часу передачі тощо.

2. Рівень передачі даних *(Data Link Layer)* або **канальний рівень** повинен забезпечити достовірну доставку даних від одного вузла до іншого в одній і тій самій мережі на фізичному рівні.

Всі дані, які передаються з верхнього (мережного) рівня, передаються у вигляді кадрів, розмір яких залежить від конкретної мережі. За допомогою спеціальних алгоритмів проводиться контроль за правильною передачею на фізичному рівні з подальшою реакцією на помилки.

Окрім підтримки каналу зв'язку та контролю за помилками, в мережах із загальним доступом до сумісно використовуваного фізичного каналу на цьому рівні регламентуються алгоритми та процедури доступу. Ця проблема вирішується введенням допоміжного підрівня – підрівня доступу до носія.

1. Фізичний рівень (*Physical Link Layer*) повинен забезпечити достовірну доставку бітів від передавача до приймача по обраному фізичному середовищу. Він описує механічні та електричні характеристики, а також визначає фізичне середовище для передачі даних. Тобто, він займається передачею та прийомом бітів по каналу зв'язку. Принциповими питаннями тут є: який тип сигналу (напруга, струм, світло тощо) і якої величини використовується для передачі логічної одиниці та нуля; тривалість одного біта (швидкість); напрямок передачі даних; коли розпочати та закінчити передачу; кількість і призначення фізичних каналів зв'язку тощо.

У комп'ютерних мережах, як правило, використовуються п'ять рівнів моделі *OSI* – без сеансового рівня та рівня подання. В промислових мережах часто відсутній опис на мережному рівні, практично завжди відсутній транспортний, а інколи обходяться навіть без прикладного. На

сьогоднішній день в стандартах *IEC* (*International Electrotechnical Commission* – Міжнародна електротехнічна комісія) визначена своя трирівнева модель архітектури промислових мереж, проте в технічній документації до опису функціонування мереж поки що прийнято давати їх архітектуру в контексті моделі *OSI*. В основному промислові мережі описують фізичний, канальний та прикладний рівні.

Приклад функціонування промислової мережі в контексті моделі *OSI* наведений у наступних розділах для мереж *ModBus*, *OWEN* та *DCON*. Але для кращого розуміння функціонування мереж розглянемо послідовність проходження даних від відправника до одержувача і визначимо допоміжні терміни, які використовуються при описі функціонування мереж.

Апаратуру або/і програму, яка займається завданнями кожного з рівнів, будемо називати *об'єктом рівня* або *сутністю рівня*. Тобто введемо такі позначення: прикладний об'єкт (прикладна сутність), транспортний об'єкт (транспортна сутність), мережний об'єкт (мережна сутність), канальний об'єкт (канальна сутність), об'єкт фізичного рівня (сутність фізичного рівня).

Нехай на *Вузлі 1* (рис. 1.18) функціонують три прикладні програми (надалі процеси) під умовними позначками *A*, *B*, *C*, а на *Вузлі 2* – процеси *D*, *E*, *F*. Процес *B* хоче обмінюватись даними з процесом *E*, для чого вони повинні скористатися правилами обміну, зрозумілими для обох, тобто визначеним *протоколом*. Частково цей протокол визначає семантику спілкування, тобто відповідність символьних позначень функціям, які необхідно виконати, та дані, які необхідно передати з цими функціями. Таким чином між двома прикладними програмами (в даному контексті на сьомому рівні *OSI*) ведеться обмін кодом та даними, за правилами протоколу даного рівня, які називають *APDU* (*Application Protocol Data Unit* – блок даних протоколу прикладного рівня). Але оскільки процеси *B* і *E* повинні якось взаємодіяти між собою, то за передачу даних між цими процесами відповідає транспортний рівень.

Для транспортного рівня код і дані прикладного рівня – це дані, які треба передати від одного процесу до іншого. Для цього необхідно використати правила адресації цих процесів, щоб правильно доставити дані,

тобто щоб їх отримав процес *E,* а не, скажімо, *F*. Ці правила, і не лише вони, визначаються протоколом транспортного, тобто четвертого рівня. Для того щоб транспортний об'єкт на *Вузлі 2* міг ідентифікувати процес, якому передають дані (в нашому випадку *E*), транспортний об'єкт на *Вузлі 1* повинен добавити допоміжні дані до даних, які вже отримав у прикладного рівня. Дані, які додаються на кожному рівні перед даними з вищого рівня, називаються **заголовком** (*Header*). Таким чином, для ідентифікації процесу одержувача транспортний об'єкт відправника додає його адресу (*E*) в заголовок *H1* і передає управління мережному рівню. Заголовок разом з даними на транспортному рівні прийнято називати *TPDU* (*Transport Protocol Data Unit* – блок даних протоколу транспортного рівня).

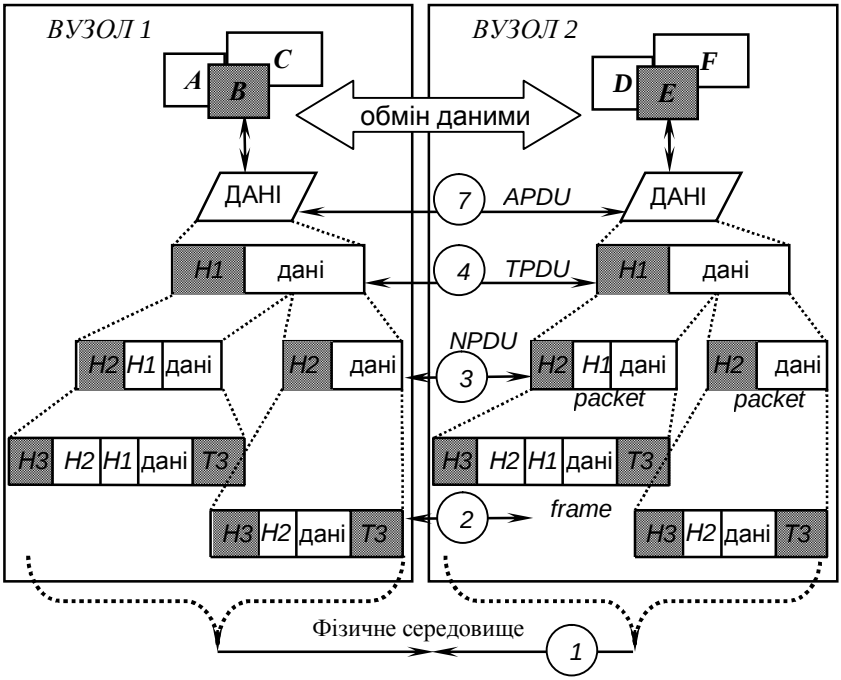


Рисунок 1.18 – Порядок обміну даними через мережу між процесами

Мережний рівень, в свою чергу, повинен забезпечити доставку блоків *TPDU* від вузла до вузла, які можуть знаходитись в різних за принципом функціонування підмережах. Для цього до *TPDU* додається заголовок з адресою приймача та відправника (в нашому випадку *H2*), який необхідний маршрутизаторам (*router*) для знаходження правильного шляху до необхідної підмережі. Окрім того, на даному рівні дані можуть бути розбиті на частини, для можливості їх передачі на нижніх рівнях (це саме може відбуватися на всіх рівнях). Ці частини разом із заголовками називаються **пакетами** (*packet*) або *NPDU* (*Network Protocol Data Unit* – блок даних протоколу мережного рівня). В нашому випадку модуль *TPDU* на мережному рівні розділений на два пакети із заголовками *H2*, які передаються послідовно один за одним канальному рівню.

Пакети, які отримує канальний рівень, вставляються в **кадри** (*frame*). Кадри мають заголовок, в якому міститься інформація про адресу вузла отримувача та/або відправника в локальній мережі та/або допоміжні службові коди. Отримувач може бути, в свою чергу, лише допоміжним пристроєм для забезпечення передачі даних в іншу підмережу, тому може і не бути кінцевим вузлом призначення. Але для канального рівня це не важливо, бо функції маршрутизації належать мережному рівню. Окрім заголовка, в кінці до пакета може додаватися **кінцевик** (*terminator*), зі службовою інформацією, наприклад даними контролю за правильністю передачі бітів. Безпосередня передача кадру по бітах забезпечується апаратно-програмними засобами фізичного рівня.

Отримання даних та їхня обробка проходить подібно до розглянутого, але навпаки. Засоби канального рівня на *Вузлі 2* в прийнятому кадрі визначають із заголовка, що кадр призначений саме їм та за кінцевиком перевіряють правильність отриманих даних, вилучають пакет та передають мережному рівню. Це саме вони роблять з наступним пакетом. Мережний рівень, в свою чергу, вилучає блоки *TPDU* з пакетів, об'єднує їх і передає транспортному рівню, який за заголовком *H1* ідентифікує процес *E* і передає йому дані.

Це, звісно, спрощений і далеко не повний перелік дій з даними на кожному рівні, але він дає уявлення про послідовність і принципи оброблення даних під час обміну інформацією між вузлами в мережі.

1.9. Поняття про промислові мережі

Як раніше було зазначено, прикладом мережі можуть бути два вузли, які з'єднані каналом зв'язку (дротовим або радіоканалом) та спільно використовують інформаційні ресурси. Унаслідок своєї апаратної і програмної структури така мережа не може забезпечувати ту швидкість і цілісність, які необхідні для надійного і високопродуктивного управління технологічним процесом і виробництвом одночасно, підтримуючи безліч користувачів та ресурсів в режимі реального часу. Ключовим рішенням вказаної проблеми може служити застосування спеціальної мережі, яка забезпечить надійну та безперервну роботу АСУ ТП і В.

У промисловості, наукових дослідженнях широко використовують як спеціальні керуючі обчислювальні комплекси, промислові комп'ютери, так і програмовані логічні контролери (ПЛК), пристрої людино-машинного інтерфейсу (ЛМІ, або англ. *HMI – Human Machine Interface*), роботу яких при необхідності координує центральний комп'ютер. У достатньо простих системах в нормальних зовнішніх умовах роботи для виконання функцій контролерів або центрального комп'ютера можуть використовуватися звичайні персональні комп'ютери. У складних умовах експлуатації застосовуються спеціальні промислові комп'ютери (*IPC – industrial PC*). Для забезпечення роботи апаратних засобів, які входять до мережі, розробляється і встановлюється спеціальне програмне забезпечення.

Система управління технологічним процесом зазвичай виконує багато різних функцій, які можна розділити на три великі групи (рис. 1.19):

- збір та оцінка даних технічного процесу – це моніторинг;
- управління деякими параметрами технологічного процесу;
- зв'язок вхідних і вихідних даних – зворотний зв'язок, тобто автоматичне управління.

Моніторинг процесу і збір інформації про процес – це основна функція, яка властива усім системам управління. Моніторинг – це збір значень змінних процесу, їх зберігання і відображення у зручній для людини-оператора формі. Моніторинг є фундаментальною властивістю усіх систем обробки даних.

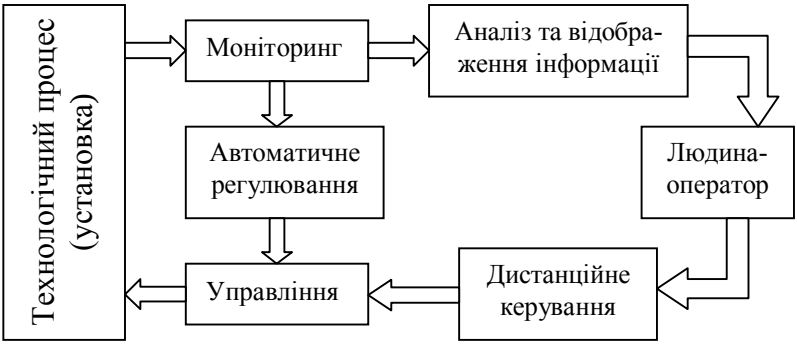


Рисунок 1.19 – Основні функції АСУ ТП і В і зв'язки між ТП (ТУ) та людиною-оператором

Моніторинг може бути обмежений лише виведенням первинних або оброблених даних на екран монітора або на папір, а може включати складніші функції аналізу та відображення. Наприклад, змінні, які не можна безпосередньо виміряти, повинні розраховуватися або оцінюватися на основі наявних (непрямих) вимірів. Іншою класичною рисою моніторингу є перевірка, що виміряні або розраховані значення знаходяться в допустимих межах.

Якщо функції системи управління технологічним процесом обмежені збором та відображенням даних, усі рішення про керуючі дії приймаються людиною-оператором. Цей вид управління, так зване супервізорне або дистанційне управління (*supervisory control*), був дуже поширений в перших системах комп'ютеризованого управління процесами. Він досі застосовується, особливо для складних та відносно від повільних процесів, де важливе втручання людини. Прикладом є біологічні процеси, де певну частину спостережень не можна виконати за допомогою автоматики без участі людини.

При надходженні нових даних їх значення оцінюється відносно допустимих меж. У розвиненішій системі контролю декілька результатів можуть комбінуватися на основі більш-менш складних правил для перевірки, чи знаходиться процес у нормальному стані або вийшов за які-

небудь допустимі межі. У ще сучасніших рішеннях, особливо побудованих на експертних системах або базах знань, комбінована оперативна інформація датчиків об'єднується з оцінками, зробленими операторами.

Управління – це функція, яка зворотна моніторингу. У прямому розумінні управління означає, що команди ПК надходять до виконавчих механізмів для впливу на фізичний процес. У багатьох випадках на параметри процесу можна впливати лише опосередковано, через інші параметри управління.

Система, яка діє автономно і без прямого втручання оператора, називається автоматичною. Система автоматичного управління може складатися з простих контурів управління (одного для кожної пари вхідних та вихідних змінних процесу) або із складніших регуляторів з багатьма входами і виходами.

Існують два основних підходи до реалізації зворотного зв'язку в обчислювальних системах. При традиційному прямому цифровому управлінні (ПЦУ, *Direct Digital Control – DDC*) центральний ПК розраховує керуючі сигнали для виконавчих механізмів. Усі ці спостереження передаються в повному обсязі від датчиків до центру управління, а керуючі сигнали – до виконавчих механізмів.

У системах розподіленого прямого цифрового управління (РПЦУ, *Distributed Direct Digital Control – DDDC*) обчислювальна система має розподілену архітектуру, а цифрові регулятори реалізовані на основі локальних процесорів, тобто розташовані поблизу технологічного процесу. ПК верхніх рівнів управління розраховують опорні значення, а локальні процесори відповідальні головним чином за безпосереднє управління технологічним процесом, тобто вироблення управляючих сигналів для виконавчих механізмів на основі даних локального моніторингу. Ці локальні регулятори включають цифрові контури управління.

З точки зору структуризації рівнів управління та обробки відмінність між прямим і розподіленим прямим цифровим управлінням полягає в тому, що в першому випадку, навіть за наявності декількох ПК, вони займаються лише передачею інформації і не приймають рішень (окрім

центрального) про керуючі. Навпаки, в розподіленій структурі на рівнях процесу, ділянки або загального управління можуть діяти більш-менш автономно і не залежать від центрального ПК. Як уже вказувалося, ця відмінність впливає і на надійність складної системи. При відмові центрального ПК система типу ПЦУ зупиняється, а розподілена система, навіть при відмові одного або декількох елементів, хоча і втратить частину функцій, але продовжуватиме роботу.

В результаті поєднання програмних і апаратних засобів автоматизації формується так звана комп'ютерно-інтегрована мережа. Апаратні засоби включають плати інтерфейсу мережі та кабелі, що їх зв'язують, а програмне забезпечення – драйвери (програми для роботи з периферійними пристроями) й операційну систему мережі, яка управляє всіма процесами. Мережа може включати сервери, робочі станції, плати інтерфейсу мережі, кабелі та спільно використовувані ресурси і периферійне обладнання (принтери, сканери тощо). При підключенні будь-якого пристрою до мережі він стає вузлом мережі і називається робочою станцією. Кожен пристрій підключається до мережі за допомогою плати інтерфейсу мережі та відповідного кабеля мережі. Може використовуватися також безпровідна технологія зв'язку: радіозв'язок або ІК-зв'язок.

Коли говорять про комп'ютерно-інтегровану мережу, важливо визначити її місце в масштабі цілого підприємства. Залежно від місця мережі в ієрархії промислового підприємства вимоги до її функціональних характеристик будуть різними.

Ієрархія АСК промисловим підприємством зазвичай подається у вигляді трирівневої піраміди:

1) рівень управління підприємством (верхній рівень);

2) рівень управління технологічним процесом (цеховий рівень);

3) рівень управління пристроями (польовий рівень).

На рівні управління підприємством розташовуються звичайні комп'ютери і файлові сервери, які об'єднані локальною обчислювальною мережею. Завдання таких систем на цьому рівні – це забезпечення візуального контролю основних параметрів виробництва, побудова звітів, архіва-

ція даних. Обсяги даних, що передаються між вузлами, вимірюються мегабайтами, а часові показники обміну інформацією не є критичними.

На рівні управління технологічним процесом (ТП) і технологічним об'єктом (ТО) здійснюється поточний контроль й управління або в ручному режимі з операторських пультів, або в автоматичному режимі за закладеним алгоритмом. На цьому рівні виконується узгодження параметрів окремих ділянок виробництва, відпрацювання аварійних і передаварійних ситуацій, параметризація і конфігурація контролерів нижнього рівня, завантаження технологічних програм (рецептів), дистанційне керування виконавчими механізмами. Інформаційний кадр на цьому рівні містить, як правило, декілька десятків байтів, а допустимі часові затримки можуть складати від 100 до 1000 мс, залежно від режиму роботи.

На рівні управління пристроями розташовуються контролери, які здійснюють безпосередній збір даних від датчиків і керування виконавчими пристроями. Розмір даних, якими контролер обмінюється з такими пристроями, зазвичай становить декілька байтів при швидкості опитування пристроїв не більше 10…50 мс.

Структура побудови мереж АСУ ТП і В в першу чергу визначається засобами і характеристиками взаємозв'язку окремих компонентів мережі (контролерів, пультів оператора, віддалених блоків введення-виведення), тобто можливостями мережі. Гнучкість і різноманітність даних мереж залежать від:

- кількості наявних рівнів мереж;
- можливих типів зв'язку (топології) на кожному рівні мережі;
- параметрів мережі кожного рівня: типів кабелю, допустимих відстаней, максимальної кількості вузлів, що підключаються до кожної мережі, швидкості передачі інформації, методів доступу компонентів до мережі (випадковий доступ за часом доставки повідомлень або такий, що гарантує час їх доставки).

Вказані властивості характеризують можливості розподілу апаратури у виробничих цехах та перенесення блоків введення-виведення безпосередньо до датчиків і виконавчих механізмів.

Одна з найпростіших структур мережі – дворівнева. Ця структура складається з польового та цехового рівнів загальної ієрархії підприємства. Всі функціональні можливості системи чітко розділені на два рівні. Перший рівень складають контролери та модулі зв'язку з технологічним об'єктом, другий – пульт оператора, який може бути представлений робочою станцією або промисловим комп'ютером, іноді панельним контролером (об'єднання в одному корпусі панелі оператора та ПЛК).

На рівні контролерів в такій системі виконується:

- збір сигналів від датчиків, встановлених на ТО;
- попередня обробка сигналів (фільтрація і масштабування);
- реалізація алгоритмів управління і формування сигналів, які управляють виконавчими механізми на ТО;
- передача та приймання інформації з промислової мережі та взаємодія з верхнім рівнем.

На рівні пульта оператора відбувається:

- формування запитів мережі до контролерів нижнього рівня;
- отримання від них оперативної інформації про хід ТП та стан ТО;
- відображення перебігу ТП та стан ТО на моніторі в зручному для оператора вигляді;
- здійснення довготривалого зберігання динамічної інформації (ведення архіву) про хід технологічного процесу та можливі події на ТО;
- корекція необхідних параметрів алгоритмів управління та уставок регуляторів в контролерах нижнього рівня.

Збільшення інформаційної потужності (кількості вхідних/вихідних параметрів) об'єкта управління, розширення кола завдань, що вирішуються на верхньому рівні управління, підвищення показників надійності приводять до появи більш складних структур мереж.

Взагалі, комунікаційна технологія побудови єдиної інформаційної мережі, яка об'єднує контролери, датчики і виконавчі механізми, визначається одним терміном «промислові мережі».

Отже, промислова мережа (дослівним перекладом англійського слова *fieldbus* є термін «польова шина») – це мережа передачі даних, яка зв'язує різні датчики, виконавчі механізми та промислові контролери та використовується в промисловій автоматизації.

Термін «промислова мережа» є найбільш адекватним перекладом. Даний термін вживається в професійній технічній літературі переважно стосовно інформаційної мережі АСУ ТП і В. Функціонально промислова мережа призначена для:

- передачі та приймання даних про параметри ТО;
- налаштування, введення в експлуатацію і діагностики обладнання АСУ ТП і В;
- живлення датчиків та виконавчих механізмів;
- передачі даних між датчиками і виконавчими механізмами, минаючи ПЛК;
- зв'язків між датчиками, виконавчими механізмами, ПЛК та АСУ ТП і В верхньої ланки;
- зв'язків між контролерами і системами людино-машинного інтерфейсу.

Промислова мережа логічно дуже схожа на звичайні комп'ютерні мережі. Проте такі мережі мають специфічний набір властивостей, що відрізняють їх від звичайних мереж:

- жорстка детермінованість (передбаченість) поведінки;
- забезпечення функцій реального часу;
- робота на довгих лініях з використанням недорогих фізичних середовищ (наприклад, скручена пара);
- підвищена надійність фізичного і канального рівнів передачі даних для роботи в промисловому середовищі (наприклад, при великих електромагнітних перешкодах та несприятливих погодних умовах);
- наявність спеціальних високонадійних механічних зєднувальних компонентів (конекторів).

Ключовими для опису промислової мережі є поняття *детермінованої* поведінки, яке припускає, що всі можливі події в мережі можуть бути наперед чітко визначені, та підвищеної *надійності* передачі даних.

Таким чином, *промислова мережа* – це, по-перше, фізичний спосіб об'єднання пристроїв і, по-друге, програмно-логічний протокол їх взаємодії. У промислових мережах для передачі даних застосовують кабелі, оптоволоконні лінії, бездротовий зв'язок. Промислові мережі можуть взаємодіяти із звичайними комп'ютерними мережами, зокрема використовувати глобальну інформаційну мережу *Internet*. В останньому випадку їх називають комп'ютерно-інтегрованими мережами.

Переваги використання промислових мереж:

- у декілька разів знижується витрата на кабелі та їх прокладення;
- збільшується допустима відстань до датчиків та виконавчих пристроїв;
- спрощується управління мережею датчиків та виконавчих механізмів;
- спрощується модифікація системи при зміні типу датчиків, протоколу взаємодії, додаванні пристроїв введення-виведення;
- є можливість дистанційно налаштовувати (конфігурувати) датчики та проводити їхню діагностику.

Проте промислові мережі мають і деякі недоліки:

- при обриві кабелю втрачається можливість одержувати дані та управляти не одним, а декількома пристроями;
- для підвищення надійності доводиться резервувати канали зв'язку.

Якщо узагальнити усі характеристики та параметри мереж (див. попередні підрозділи), то промислова мережа є цифровою, двонаправленою, багатоточковою, послідовною комунікаційною мережею, яка використовується для зв'язку ізольованих один від одного (за функціями) таких пристроїв, як контролери, датчики, силові приводи тощо. Кожен *field*-пристрій володіє самостійним обчислювальним ресурсом, що дозволяє відносити його до розряду інтелектуальних (*SFD – Smart Fieldbus Device*). Кожен такий пристрій здатний самостійно виконувати ряд функцій з самодіагностики, контролю та обслуговування функцій двонаправленого зв'язку. Доступ до нього можливий не лише з боку контролера, але і з боку аналогічних йому пристроїв. Тому технологія такої мережі

розширює можливості технології використання уніфікованих струмових сигналів 4…20 мА.

На сьогоднішній день споживачі поступово відходять від практики застосування власних і централізованих систем управління та починають звертати увагу на системи з розподіленим інтелектом. Основна мета побудови *розподілених* систем автоматизації – це здешевлення та спрощення технологій і менеджменту виробництва й експлуатації кінцевої системи за рахунок забезпечення технології наскрізного доступу до мережі: від могутніх супервізорних комп'ютерів і багатофункціональних контролерів до інтелектуальних елементів (датчики, регулятори тощо). При цьому такий зв'язок повинен задовольняти всім сучасним вимогам за функціональністю, надійністю, захищеністю та відкритістю.

У результаті фірмова і централізована архітектура здає свої позиції на ринку, тоді як *відкриті розподілені системи* (РСУ), в яких для управління, збору даних та обміну інформацією використовуються промислові мережі, починають його завойовувати. Одна з причин цього криється в тому, що прокладка кабелів і розгортання системи з використанням промислових мереж обходиться значно дешевше. Системи з централізованим управлінням зазвичай вимагають, щоб кожен датчик або група датчиків підключалася до центрального контролера окремим (часто дорогим) високоякісним кабелем. Навпаки, в системі на базі промислової мережі, поряд з кожним кластером датчиків, розташовується один інтелектуальний вузол, що перетворює сигнали датчиків в цифрову послідовність і передає їх у систему управління/моніторингу.

Щодо реалізації промислових мереж стосовно моделі *OSI* є деякі обмеження:

- оскільки у більшості промислових мереж відсутній транспортний рівень, то максимальний розмір повідомлень прикладного рівня обмежений максимальним розміром, який допускає канальний рівень;
- відсутність мережного рівня зумовлює неможливість маршрутизації між різними мережами;

- відсутність сеансового рівня не дозволяє використовувати повнодуплексний режим;
- в мережах, де відсутній рівень подання, формати та синтаксис повідомлень повинні бути однаковими для всіх пристроїв.

Окремо потрібно зазначити, що невідповідність прикладного рівня промислових мереж стандартній моделі *OSI* потребує введення додаткового рівня, який має назву «рівень користувача».

Хоча *fieldbus*-технології з'явилися вже близько 25 років тому, абсолютно домінуючими вони ще не стали. Це пов'язано в основному з відсутністю єдиного міжнародного стандарту на протокол промислової мережі, який міг би гарантувати повну взаємозамінюваність і сумісність між виробами різних виробників.

На завершення розділа можна зробити такі висновки:

1) Можна констатувати, що ринок промислових мереж сформувався: існує велика пропозиція технологій мереж, величезний спектр готових виробів, провідні компанії об'єднані в різні асоціації і групи, постійно ведуться роботи із стандартизації як в рамках національних, так і міжнародних комітетів із стандартизації.

2) *Fieldbus*-технології – це шлях до припинення протистояння виробників обладнання для систем автоматизації виробництв, з одного боку, та універсальні інструменти для побудови інтегрованих комплексів – з іншого.

3) При виборі комунікаційної технології можна керуватися кількісними параметрами (обсяг корисних даних, що передаються, максимальна довжина шини, допустима кількість вузлів на шині, перешкодозахищеність та ін.), ціновим критерієм (витрати з розрахунку на один вузол), популярністю, ефективністю рішення задачі, простотою конфігурації тощо. При цьому поліпшення одного параметра може привести до погіршення іншого. Тому при виборі того або іншого протокольного рішення необхідно слідувати принципу розумної достатності.

4) Найбільш важливим критерієм вибору повинна бути відповідність принципам «*відкритих систем*» – стандартизація і доступність.

Лише це дозволяє і виробникам, і користувачам робити надійні прогнози та гарантувати збереження їх інвестицій.

5) Кожна реалізація промислової мережі має свої переваги і недоліки. Важливо визначити деяку підмножину рішень для того, щоб сконцентрувати на ній основні зусилля і виробників устаткування, і розробників супутного програмного забезпечення, і системних інтеграторів.

Контрольні запитання:

1) У чому полягає сутність поняття «інформація»? Наведить приклади.
2) Які функціональні елементи складають узагальнену інформаційну систему? Дайте коротку характеристику.
3) Яке призначення промислових мереж у складі АСУ ТП? Переличіть їхні переваги відносно звичайних комп'ютерних мереж.

РОЗДІЛ 2.
ОСНОВИ ПОБУДОВИ ТА ВИКОРИСТАННЯ ПОСЛІДОВНИХ ІНТЕРФЕЙСІВ В ПРОМИСЛОВИХ МЕРЕЖАХ

У цьому розділі розглянуто теоретичні та практичні питання побудови та використання послідовних інтерфейсів, які є апаратною основою інформаційно-управляючих систем. Наведено інформацію про побудову симетричних і несиметричних ліній передачі даних та опис роботи перетворювачів послідовних інтерфейсів.

Так, завершивши вивчення цього розділу, ви зможете:

- отримати уявлення про принципи реалізації обміну даними по послідовних інтерфейсах *RS-232* та *RS-485*;
- ознайомитись з побудовою та принципом дії універсального асинхронного приймача та передавача;
- оволодіти знаннями про стандарти, які описують послідовні інтерфейси;

- описати принцип дії перетворювачів послідовних інтерфейсів;
- налагоджувати апаратно-програмні засоби систем обміну даними, які використовують послідовні інтерфейси.

2.1. Установи зі стандартизації послідовних інтерфейсів

Необхідність стандартизації інтерфейсів між різними компонентами і системами давно усвідомлена і промисловцями, і користувачами. В першому розділі посібника вже згадувались деякі стандарти та посилання на них.

Основне призначення стандартів – це забезпечення сумісності різних компонентів і необхідної якості продукції. Стандарти розробляються національними і міжнародними комітетами, а також численними професійними та урядовими організаціями, в роботі яких зазвичай беруть участь представники компаній-виробників, наукових організацій, університетів, урядових установ. Розроблені ними документи використовуються як керівництво при створенні нових компонентів та наданні послуг. Проте, у багатьох випадках діючі стандарти не обов'язково є кращими технічними рішеннями, оскільки вони описують вже існуючі реалізації і таким чином створюють «зони сумісності». Іноді організації зі стандартизації намагаються визначити загальні рамки, на базі яких повинні створюватися конкретні стандарти і технічні рішення.

З іншого боку, деякі розробки окремих компаній отримали таке широке схвалення в промисловості, що стали стандартами *де-факто*. В цьому відношенні «анархія», властива промисловості і вільному ринку, дає кращі результати, які хоча і не є стандартами в повному розумінні, але мають більше поширення і визнання, ніж те, що створюється заорганізованою і повільною міжнародною бюрократією. Найбільш успішними виявилися ті рішення, які залишають достатню свободу виробникам і дозволяють випускати продукцію з розумними витратами на базі наявних механічних та електричних розробок.

У різному контексті використовується різна термінологія – «стандарти» (*standards*), «рекомендації» (*recommendations*), «керівні вказівки» (guideunes) тощо. Офіційний статус стандартів мають лише ті документи, які випущені

урядовими або загальновизнаними професійними установами. Будь-які інші рекомендації, керівництва й інші документи не носять офіційного характеру і їх застосування залежить виключно від користувача.

Стандарти можуть бути або офіційними – *де-юре*, або фактичними – *де-факто*. Офіційні (*legal*) стандарти випускаються або державними, або міжурядовими організаціями, або загальновизнаними організаціями, які створені виробниками. Фактичні стандарти, так звані промислові, виникають досить спонтанно, коли на практиці майже усі роблять одне і те ж або використовують один і той же продукт, але офіційно ніхто не формулює відповідне рішення. Так було, наприклад, з «промисловим стандартом ПК», що означало комп'ютер «*IBM PC*» та його клони, з модемним протоколом компанії *Hayes*, мовою управління принтером компанії *Epson* й іншими. Багато фактичних стандартів згодом було схвалено офіційними установами. Характерно, що фактичними стандартами користуються усі, хоча ніхто не зобов'язує це робити.

Найбільш важливою міжнародною організацією зі стандартизації є *International Organization for Standardization* (*ISO*). Скорочена назва походить до грецького кореня «*isos*», що означає «рівний», і це дозволяє використовувати його в різних мовах незалежно від того, як звучить в них повна назва і відповідна абревіатура цієї організації. Наприклад, по-французьки вона називається *Organisation Internationale de Normalisation*, а по-українськи – Міжнародна організація зі стандартизації. До складу *ISO* входять близько 90 національних комітетів й організацій із стандартизації, включаючи *ANSI* (США), ДСТУ (Україна), *DIN* (Німеччина), *AFNOR* (Франція), *BSI* (Великобританія), *UNI* (Італія), *SS* (Швеція) та інші, які випускають стандарти для усіх галузей і, зокрема, для промисловості та обчислювальної техніки. В принципі, усі країни-учасниці повинні дотримуватися стандартів *ISO*. З іншого боку, деякі національні стандарти виявилися настільки вдалими, що вони цілком «імпортувалися» іншими країнами; наприклад, так було з німецькими стандартами *DIN* щодо шини *Profibus*.

Підрозділ *ISO*, який розробляє стандарти в галузі електротехніки і електроніки, – це *International Electrotechnical Commission* (*IEC*, Міжна-

родна електротехнічна комісія – МЕК, http://www.iec.ch). До складу МЕК входить технічний комітет *TC65*, який займається питаннями вимірів та управління в промислових процесах (*Industrial Process Measurement and Control*). Комітет *TC65*, у свою чергу, складається з декількох робочих груп, кожна з яких відповідає за найбільш важливі напрями технології управління процесами, від системних питань і програмного забезпечення до програмованих логічних контроллерів і цифрових комунікацій. Технічний комітет *TC61* займається питаннями використання ПЛК для управління технологічними процесами та розроблення ППЗ для ПЛК.

Країни, члени Європейського Союзу (ЄС), окрім участі в *ISO*, мають в його структурі власні організації – *Comité Européen de Normalisation* (*CEN*, Європейський комітет із стандартизації) і *Comité Européen de Normalisation Electrotechnique* (*CENELEC*, Європейський комітет із стандартизації в галузі електротехніки). Ці комітети були організовані в 1960-х роках, щоб продовжувати роботи *ISO* і МЕК на європейському рівні. Комісія Європейського Союзу наділена правом вводити єдині стандарти у рамках політичної та економічної інтеграції, тому останніми роками діяльність *CEN* і *CENELEC* набула особливого значення.

У галузі телекомунікацій і зв'язку найважливіше значення мають рекомендації *International Telecommunications Union* (*ITU*, Міжнародний союз електрозв'язку – МСЕ,) і його підрозділи *International Telecommunications Union – Telecommunication Standardization sector* (*ITU – T*, Міжнародний союз електромережі – сектор телекомунікацій), яке раніше називалося *Comité Consultatif International de Télégraphie et de Téléphonie* (*CCITT*, Міжнародний консультативний комітет з телеграфії і телефонії – МККТТ). Досі багато рекомендацій останнього зберегли в назві абревіатуру *CCITT* (у російській нотації – МККТТ). Усі країни-члени Організації Об'єднаних Націй – представлені в *ITU* державними телекомунікаційними компаніями, за винятком США, де немає державної монополії на послуги зв'язку, і тому їх представляє Держдепартамент (тобто Міністерство закордонних справ США).

Організація *ITU* випускає не стандарти, а лише рекомендації. В принципі, кожна національна компанія може діяти на території своєї кра-

їни за власним розсудом, але очевидно, що для міжнародного телефонного зв'язку і передачі даних потрібна сумісність національних стандартів. Тому в цій сфері існує тенденція до вироблення загальних правил. Наприклад, цифрові мережі з інтеграцією послуг впроваджуються у всьому світі відповідно до однотипних вимог.

Стандарти *ISO* охоплюють усі галузі техніки і тому разом з відповідними національними стандартами є основними для промисловості. Рекомендації ITU стосуються, в першу чергу, передачі голосу і даних. Якщо сфери дій цих організацій перетинаються, наприклад, при визначенні комунікаційних інтерфейсів або устаткування інформаційних мереж, загальний стандарт публікується двічі від імені кожної з них і має різні позначення.

У США існує *Electronics Industries Alliance* (*EIA*, Асоціація електронної промисловості), яка об'єднує виробників електронного устаткування і розробляє відповідні електричні і функціональні стандарти. Документи, що випускаються цією асоціацією, мають ідентифікатор *RS* або *EIA*. Деякі з розробок *EIA* отримали широке поширення за межами США, зокрема інтерфейс комунікаційного порту *RS-232*, який зараз також називають *EIA-232*.

Розташований в США *Institute of Electrical and Electronics Engineers* (*IEEEE*, Інститут інженерів з електротехніки і радіоелектроніки – IIEP) – це професійне об'єднання технічних фахівців більш ніж 100 країн, що має національні відділення по всьому світу. Ця організація випускає свої власні стандарти, що стосуються принципів роботи, продуктивності і якості електротехнічного та електронного устаткування. Стандарти *IEEE* визнані у всьому світі, а в США вони випускаються спільно з *American National Standards Institute* (*ANSI*, Американський національний інститут стандартів).

Отже, підсумуємо. Існує сім основних міжнародних організацій, які займаються розробленням стандартів та виробленням рекомендацій, що пов'язані з передаванням даних:

- *ISO*: Міжнародна організація із стандартизації;
- *ITU-T*: Міжнародний телекомунікаційний союз;

- *IEEE*: Інститут інженерів з електротехніки та електроніки;
- *IEC*: Міжнародна електротехнічна комісія;
- *RS*: Асоціація електронної промисловості;
- *ANSI*: Національний інститут стандартизації США;
- *TIA*: Асоціація промислових засобів зв'язку.

Організація *ANSI* є головною організацією зі стандартизації США і представляє цю країну в *ISO*. *ANSI* – некомерційна, неурядова організація, яка підтримується більш ніж 1000 торговельними об'єднаннями, професійними співтовариствами та компаніями.

Міжнародний союз телекомунікацій (*ITU*) є спеціалізованим агентством Організації Об'єднаних Націй (*UNO*). Він складається з представників від поштово-телеграфних і телефонних організацій, користувачів і виробників телекомунікаційного обладнання. В Європі виконавча влада має тенденцію суворо дотримуватися рекомендацій *ITU*. Хоча в минулому виробники США їх не визнавали, тепер вони починають швидко пристосовуватися до рекомендацій *ITU*.

ITU визначає повний набір стандартів, що стосуються взаємодії телекомунікаційного обладнання. Стандарти на обладнання обміну даними зазвичай визначаються низкою рекомендацій *ITU-T «V»*.

Два приклади стандартів фізичних інтерфейсів *ITU-T*:

- *V.24*: еквівалент *RS-232* для повільних асинхронних послідовних каналів;
- *V.35*: еквівалент *RS-449* для широкосмугових каналів.

RS є громадською організацією зі стандартизації в США, що спеціалізується на електричних та функціональних характеристиках інтерфейсного обладнання. Вона представляє виробників електронного устаткування. Оскільки *RS* і *TIA* в 1988 році злилися, то *TIA* представляє телекомунікаційний сектор *RS*, і ці літери є на деяких документах зі стандартизації.

IEC є міжнародною організацією зі стандартів, яка підпорядкована *ISO*. Її діяльність зосереджена на електричних стандартах. *IEC* виникла в Європі та її стандарти використовуються в більшості західних держав, за винятком США, і близько пов'язаних з ними країнах.

IEEE є професійним співтовариством інженерів з електротехніки та

електроніки США. Воно випускає свої власні стандарти і коди. *IEEE* є членом *ANSI* і *ISO*.

ISO залучає членів з усіх країн світу і концентрується на міжнародній координації стандартів.

2.2. Загальна характеристика апаратних засобів послідовних інтерфейсів

Реалізація передавання бітів на фізичному рівні, а саме формування сигналів, здійснюється за допомогою спеціальних інтегральних мікросхем, які є частиною інтерфейсу між шиною мікропроцесора та лінійним формувачем (приймачем) каналу зв'язку. Такі мікросхеми мають назву *USART* (*Universal Sinchronous/Asynchronous Receiver-Transmitter* – універсальний синхронно-асинхронний приймач-передавач) і є досить простими у виробництві та поширеними в засобах зв'язку.

Вихід *USART* безпосередньо не підключається до каналу зв'язку, а видає біти на рівні *TTL* (3,3 … 5 В) на вхід трансивера. На рис. 2.1 показаний приклад використання *USART* разом з трансивером *RS-232*. Різні схеми використовуються також в синхронній передачі даних і називаються *USRT*.

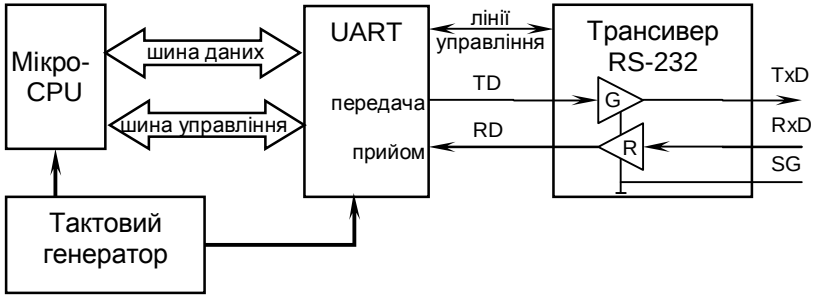


Рисунок 2.1 – Приклад використання *UART* з *RS-232*

При передачі *UART* послідовно виконує такі дії:

- встановлює для трансивера бітову швидкість;
- приймає від мікропроцесора біти даних символу через паралельну шину;

- генерує для трансивера стартовий біт;
- послідовно передає трансиверу біти даних;
- визначає та передає трансиверу біт парності;
- передає трансиверу стопові біти;
- сигналізує процесору про готовність до передачі наступного символу;
- при необхідності координує квітування.

При прийомі від трансивера *UART* послідовно виконує такі дії:

- встановлює для трансивера бітову швидкість;
- розпізнає стартовий біт;
- зчитує біти даних, що надходять від трансивера у вигляді послідовності бітів;
- зчитує біт парності та робить перевірку на парність;
- передає символ в паралельному коді мікропроцесору;
- при необхідності координує квітування.

Крім сигналів передачі (*TD*), прийому (*RD*) та шини даних, у схемі використання *UART* також визначені додаткові управляючі сигнали для синхронізації роботи мікропроцесора з трансивером. Застосування *UART* усуває необхідність програмування в мікропроцесорі описаних вище процедур. Замість цього вони реалізуються апаратно в *UART*. Програмно реалізується лише передавання та прийом послідовності біт з/до *UART*.

Байт, який прийнятий від мікропроцесора на передавання, заноситься до пам'яті введення/виведення схеми *UART*. Біти, які необхідно передати, завантажуються до регістра зсуву, а далі вони виштовхуються негативними перепадами тактових імпульсів. Частота тактових імпульсів визначає швидкість передавання даних. У разі, коли усі біти будуть відправлені, процес повторюється для передавання іншого пакета (*символу*). Термін «*пакет*» використовується для позначення сукупності стартового біта, бітів даних, біта паритету та стопових бітів. Іноді в літературі зустрічається позначення *SDU* (*Serial Data Unit* – блок послідовних даних).

Приймальна частина *UART* постійно слідкує за приймальною лінією, очікуючи прихід стартового біта. Якщо прийнятий стартовий біт, починається приймання інших бітів з визначеною швидкістю передавання бітів та

занесення їх до регістру зсуву приймача. Після приймання усього пакета здійснюється його розшифрування та передавання паралельним кодом бітів даних до мікропроцесора.

У процесі оброблення пакета в *UART* можуть виникати типові помилки:

- переповнення буфера приймача, якщо швидкість приймання не відповідає швидкості передавання бітів;
- невідповідність біта паритету;
- помилка кадру, якщо прочитані біти не відповідають формату кадру;
- помилка розірвання зв'язку, якщо час стартового біта більше часу передавання одного кадру.

Сучасні *UART* мають буфер збільшеної ємності, що дозволяє істотно збільшити швидкість передавання даних. На цей час можлива швидкість передавання даних досягає 115200 бодів.

2.3. Симетричні та несиметричні лінії передачі

Стандарт на послідовний інтерфейс обміну визначає електричні та механічні параметри, які дозволяють устаткуванню різних виробників підключатися і взаємодіяти одне з одним. На цей час серед інтерфейсів у промислових мережах найбільш популярні декілька стандартів. Всі вони належать до стандартів послідовної передачі даних. Це стандарти: *RS-232*, *RS-485* та спеціалізований стандарт інтерфейсу – струмова петля 20 мА. Але останній інтерфейс на цей час використовують рідше.

Як лінії передачі (канали зв'язку) в стандартах застосовують симетричні та несиметричні канали.

Несиметричний канал зв'язку

У системі, яка використовує несиметричні лінії передачі, спільний опорний провідник сигналу одночасно використовується багатьма сигналами та електронними схемами. Сигнал у вигляді напруги передається лише по одному дроту, який вимірюється відносно спільного дроту, так званою сигнальною «*землею*». Сигнал, який передається, є напругою між сигнальним та спльним дротом.

Теоретично несиметрична передача повинна працювати добре, якщо струми сигналу невеликі і спільний дріт має низький опір (імпеданс). На практиці несиметричні системи працюють лише на коротких каналах зв'язку. Спільний дріт сигналу має характеристики, аналогічні іншим провідникам (опір, індуктивність і ємність), та неідеальний для використання як базовий провідник. Для довгих відстаней спільний провідник може не мати однакового потенціалу у всіх його місцях або на кінцях. На спільний провідник можуть також впливати перешкоди або паразитна напруга. Іноді як загальний дріт використовується екрануючий провідник. Це може привести до дуже великого рівня перешкод і цього слід уникати. Несиметрична передача використовується в інтерфейсах *RS-232* і *RS-423*. Той факт, що на спільний опорний провідник можуть впливати різні перешкоди, означає, що до напруги, виміряної біля приймача, можуть додаватися наведення.

Симетричний канал зв'язку

Для симетричної передачі інтерфейси вимагають для кожного сигналу по два провідники. Напруга на приймаючому кінці є різницею напруги між цими двома дротами. Така система передачі називається симетричною або диференціальною і дозволяє усунути багато проблем наведень, пов'язаних з використанням загального опорного дроту.

Симетрична лінія передачі забезпечує вищу швидкість передачі даних на довгі відстані. Диференціальний спосіб передачі даних є переважним в промисловому оточенні, де перешкоди можуть бути основною проблемою. Недолік цього способу полягає у тому, що для передачі одного сигналу симетрична система вимагає двох дротів. Успішна передача сигналу напруги по двох дротах у присутності перешкод або падіння напруги основана на тому припущенні, що ці дроти мають однакові характеристики, тому дія на них виявлятиметься однаковою. Але це не означає, що перешкоди не впливають на симетричну диференціальну систему. Напруга на обох дротах збільшуватиметься або зменшуватиметься одночасно, а різницевий сигнал при цьому залишається однаковим. Напруга між сигнальним і загальним провідниками називається синфазною напругою. Синфазна напруга є показником наведеної напруги або пере-

шкод на канал зв'язку. В ідеальному випадку синфазна напруга на двох проводах буде повністю зведена нанівець. Проте, чим більше синфазна напруга, тим вище вірогідність спотворення вихідної напруги або псування пристрою. Приймальна схема 2-провідникової диференціальної системи розроблена так, щоб ігнорувати або послаблювати синфазну напругу, використовуючи спеціальні способи придушення синфазного сигналу. Симетрична передача використовується в більшості швидкодіючих інтерфейсів, таких як *RS-422* і *RS-485*.

2.4. Загальна характеристика стандартів послідовних інтерфейсів

2.4.1. Стандартний інтерфейс RS/TIA-232 (CCITT V.24)

Стандарт інтерфейсу *RS-232C* вперше опублікований асоціацією *EIA* [4] в 1969 році як варіант «*C*» рекомендованого стандарту (*RS – Recommended Standard*) за номером «232». Інтерфейс призначений для підключення апаратури, яка передає (або приймає) дані апаратури передавання даних – АПД (*DTE – Data Terminal Equipment*) до (від) апаратури каналів даних – АКД (*DCE – Data Communication Equipment*). Розділення апаратури на *DTE* та *DCE* дозволило стандартизувати відповідні пристрої різних виробників для організації обміну даними в інформаційних мережах. Стандарт *RS-232C* складається з трьох основних частин:

- електричні параметри сигналу;
- механічні характеристики інтерфейсу;
- функціональний опис сигнальних та управляючих ліній.

Найпростіший приклад організації каналу зв'язку – це пряме з'єднання двох *DTE*, тобто двох ПК, яке зображене на рис. 2.2. Приклад системи, коли функцію *DTE* виконує комп'ютер, в якому *RS-232* реалізований як *COM*-порт, а функцію *DCE* – модем, що показаний на рис. 2.3.

Інтерфейс *RS-232* належить до послідовних каналів зв'язку. Це означає, що дані (інформація) передаються послідовно, біт за бітом по одному дроту (на відміну від паралельного інтерфейсу, в якому, наприклад, кожен біт байта передається по окремому дроту, тобто байт передається по восьми дротах). Формат кадра – один байт даних і декілька службових бітів, деякі з яких можуть бути відсутніми. В цілому стандарт визначає

електричні та механічні параметри інтерфейсу. Для передавання одного біта використовується несиметричний метод, тобто інформаційний сигнал передається за допомогою потенціалу в інформаційній лінії відносно спільного дроту. При цьому обмін інформацією між ПК та периферійним пристроєм (модем або інший ПК) по інтерфейсу *RS-232* є двостороннім, тобто дані можуть передаватися від ПК в модем і прийматися зворотно одночасно по різних лініях. Такий режим обміну називається *дуплексним* (див. рис. 1.3).

Рисунок 2.2 – З'єднання двох комп'ютерів
без використання модемів

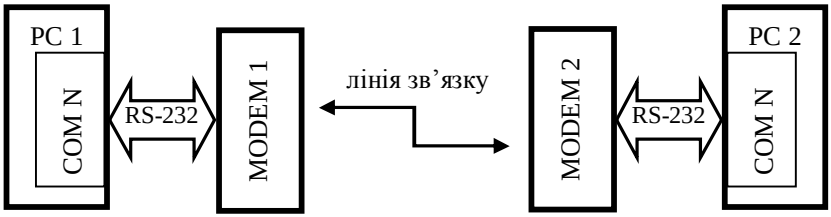


Рисунок 2.3 – Схема з'єднання двох комп'ютерів
з використанням модемів

Для передавання окремих бітів в стандарті *RS-232* передбачені електричні параметри передавачів та приймачів, функцію яких виконує стандартна мікросхема *UART*, яка є частиною інтерфейсу між шиною мікропроцесора та трансивером каналу зв'язку. Тобто вихід *UART* безпосередньо не підключається до каналу зв'язку, а видає біти на рівні *TTL* на вхід трансивера, який, в свою чергу, формує сигнали для передавання у лінію зв'язку. В табл. 2.1 наведено електричні параметри сигналів, що формуються в лінії зв'язку передавачем та отримуються на вході приймачем.

Значення параметрів управляючих ліній співпадають з параметрами інформаційних ліній, але в інверсному розумінні. Це означає, що управляючий сигнал, який має статус «TRUE», передається позитивним потенціалом. В інтерфейсі *RS-232* можливе пряме використання *TTL*-логіки, але це не завжди зручно.

Таблиця 2.1 – Електричні параметри стандарту *RS-232*

Параметр інтерфейсу	Значення, В
Напруга на виході передавача:	
логічна одиниця	-5…-25
логічний нуль	+5…+25
Напруга на вході приймача	±3…25
Невизначеність логічного рівня	±3

У стаціонарному ПК на материнській платі розміщений спеціальний з'єднувач типу *DB-9(M),* так званий комунікаційний порт (*COM*-порт). До з'єднувача за допомогою кабелю підключається інший ПК (ПЛК) або модем. Для першого випадку (див. рис. 2.2) використовують так званий «нульмодемний» кабель, що забезпечує зв'язок пристроїв типу *DTE – DTE*. Тому, для випадку прямого з'єднання двох ПК «нуль-модемний» кабель з обох боків повинен мати з'єднувач типу *DB-9(F).*

Для зв'язку ПК та модема, тобто *DTE–DCE*, застосовують так званий «прямий» кабель. Так, пристрої *DTE* мають з'єднувачі типу *DB-9(M),* а пристрої *DCE* – з'єднувачі типу *DB-9(F).* Це означає, що кабель з'єднання ПК та модема має з боку ПК з'єднувачі типу *DB-9(F),* а з боку модема – з'єднувачі типу *DB-9(M).*

Далі модем підключається до лінії фіксованого телефонного зв'язку. На цей час для диспетчеризації та моніторингу автономних пунктів все більше застосовують безпровідні модеми, тобто *GSM*-модеми або радіомодеми. В останньому випадку використовується звичайний так званий «прозорий» радіоканал.

До складу інтерфейсу входять інформаційні та управляючі лінії (дроти), які називають лініями інтерфейсу. Управляючі лінії потрібні для

управління потоком даних по інформаційних лініях. Склад та призначення цих ліній залежить від умов з'єднання. Також є певні вимоги до обладнання та типів з'єднувачів. Взагалі *DTE* відправляє дані на контакт «2» і приймає на контакт «3», а *DCE* відправляє дані на контакт «3» і приймає на контакт «2». У табл. 2.2 наведено склад і призначення ліній в інтерфейсі *RS-232* для з'єднувача типу *DB-9*. На рис. 2.4 показано з'єднувач *DB-9* з позначенням сигнальних та управляючих ліній.

Таблиця 2.2 – Позначення входів та виходів з'єднувача DB-9 (з боку термінального пристрою)

Назва сигналу	Позначення сигналу	Призначення сигналу	Напрям сигналу	Номер лінії
Signal Common	*GND*	Загальний сигнальний дріт	–	5
Transmitted Data	*TxD*	Передача даних. Вихід передавача	*DTE→DCE*	3
Received Data	*RxD*	Прийом даних. Вхід приймача	*DTE←DCE*	2
Request to send	*RTS*	Запит термінала дозволу на передачу даних	*DTE→DCE*	7
Clear to send	*CTS*	Модем вільний для передачі даних	*DTE←DCE*	8
DCE Ready	*DSR*	Вказує на готовність модема до роботи	*DTE←DCE*	6
DTE Ready	*DTR*	Готовність термінала до обміну даними	*DTE→DCE*	4
Received Line Signal Detector	*DCD*	Показує наявність сигналу несучої на вході модема	*DTE←DCE*	1
Ring indicator	*RI*	Запит на установку з'єднання від модема (прийом модемом виклику)	*DTE←DCE*	9

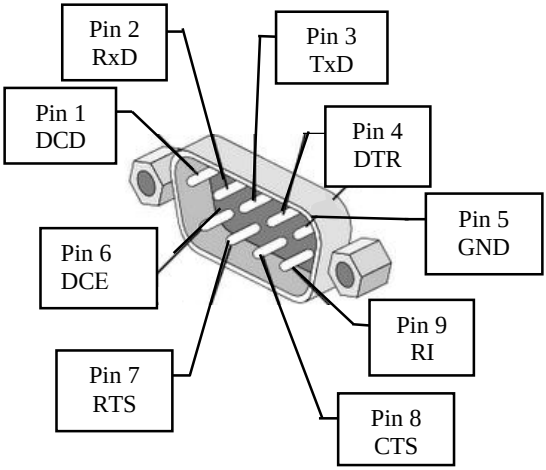
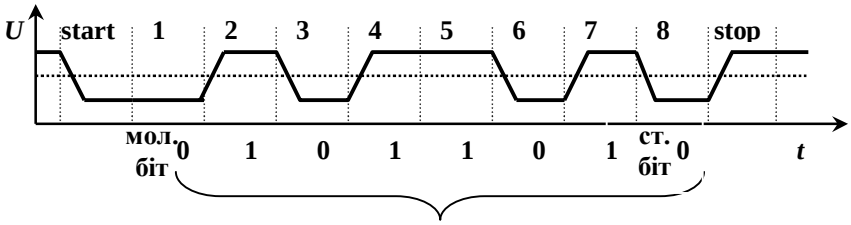


Рисунок 2.4 – Позначення сигнальних та управляючих ліній з'єднувача DB-9(M) (з боку термінального пристрою)

Для управління потоком даних в *RS-232* передбачено два способи: апаратний та програмний. Якщо використаний мінімальний «нульмодемний» кабель, тобто немає управляючих ліній, тоді обмін даними здійснюється без управління потоком даних. Для реалізації управління потоком даних апаратним способом можливі два варіанти залежно від обраних управляючих ліній. Частіше використовують пару сигналів *RTS–CTS*, що означає управління обміном з боку термінала. Рідше пару *DSR–DTR*, де обмін ведеться за ініціативою модема. В обох випадках ведений пристрій відповідає готовністю шляхом включення відповідної лінії. Програмне управління потоком даних здійснюється за допомогою використання додаткових спеціальних *ASCII*-символів, які передаються по інформаційних лініях та мають назву «*XON*» та «*XOFF*».

На рис. 2.5 показана діаграма передавання одного символу *ASCII*-коду (наприклад, символ «***Z***»=$5A_{hex}$=0101_1010_2), який передається по інформаційній лінії даних *TXD* інтерфейсу *RS-232*. Параметри кадру такі: 8 біт даних, без перевірки на парність, один стоп-біт. Як випливає з рисунка, передача починається з так званого «старт-біта», потім йдуть біти

даних, починаючи з молодшого (їх може бути від п'яти до восьми), далі слідує біт паритету або парності (який може бути відсутнім) і потім слідують стоп-біти (їх може бути або один, або два залежно від наявності біта парності). На практиці використовують вісім бітів даних, при цьому біт паритету (на рисунку не показаний), як правило, не використовується, а при максимально високій швидкості передачі бажано передавати два стоп-біти.



ASCII-символ «Z» ($5A_{hex}$)

Рисунок 2.5 – Діаграма передавання одного символу

При формуванні кадру можуть додаватися різні комбінації кількості бітів паритету та стопових бітів. При прийомі кадрів службові біти автоматично видаляються. Зазвичай стартовий та стоповий біти обрамляють один байт інформації (8 біт), проте в полі даних може бути 5, 6, 7 або 9 бітів. Для відокремлення кадру від наступного використовують 1, 1.5 або 2 стоп-біти, що означає інтервал часу, який дорівнює величині, зворотній швидкості передавання даних. Два стопових біти використовують для зменшення вірогідності розузгодження приймача та передавача при щільному трафіку, тобто високій швидкості передавання даних. Приймач ігнорує другий стоповий біт, сприймаючи його як коротку паузу в лінії.

Біт парності призначений для контролю за правильністю передачі даних і вибирається з ряду: парний (*even*), непарний (*odd*), відсутній (*none*). При виборі останнього на фізичному рівні не проводиться контроль помилок. Якщо використовується *парний* (*even*) біт паритету, то при передачі підраховується кількість одиничних бітів даних, і якщо їхня кількість непарна, то додається біт паритету, рівний логічній «1», в протилежному випадку додається логічний «0». При *непарному* (*odd*) паритеті

– навпаки, передавач буде добавляти до бітів даних логічну «1» або «0» так, щоб сума одиничних бітів у бітах даних разом з паритетним була непарною. Приймач перевіряє суму прийнятих одиничних бітів даних та паритету і якщо вона не співпадає з типом наперед визначеного біта паритету, сигналізує про це верхнім рівням, які вирішують можливість повторного запиту.

Оскільки службові біти займають частину бітового потоку, то результуюча пропускна здатність не дорівнює швидкості з'єднання. Наприклад, для 8-бітових посилок формату «*8-N-1*» синхронізуючі біти займають 20 % потоку, що для фізичної швидкості *115200* бодів дає бітову швидкість даних *92160* біт/с або *11520* байт/с.

Насамкінець відзначимо, що інтерфейс *RS-232* належить до з'єднань типу «точка-точка» і забезпечує зв'язок на відстанях не більш ніж *15* м на невеликих швидкостях і *1,5* м на швидкості *115200* бод. Такі параметри зумовлені електричними параметрами стандарту та якістю провідників у кабелі.

2.4.2. Стандартний інтерфейс RS/TIA-485

Іноді використання інтерфейсу *RS/TIA-232* не відповідає поставленим з боку АСУ ТП і В вимогам щодо обміну даними. Це стосується в першу чергу обмеження на дальність дії лінії зв'язку (до 15 м) та на модель обміну даними («точка–точка»). Такі показники не завжди відповідають умовам промислового виробництва. Ці недоліки інтерфейсу можуть бути усунені, якщо застосувати інший інтерфейс зв'язку, в якому використаний симетричний канал зв'язку (див. стор. 68). На початку 70-х років був розроблений стандарт на інтерфейс *RS/TIA-422*, який використовував 4-провідну лінію зв'язку. Цей стандарт забезпечив достатньо велику швидкість та зменшив залежність від перешкод, які пов'язані з використанням загального дроту, як це було в *RS/TIA-232*. У результаті збільшилась довжина лінії зв'язку до 1200 м, швидкість – до 10 Мбіт/с. При цьому при наявності одного лінійного формувача (передавача) в лінії можливе підключення до 10 приймачів. Тобто реалізована схема обміну «точка–багатоточка» з дуплексним каналом. Але цей стандарт не набув значного поширення серед виробників апаратури зв'язку та контролерів.

Тому логічним продовженням стала поява нового стандарту *RS/TIA-485* [5]. Цей стандарт є розширенням стандарту *RS/TIA-422*, який описує обмін по диференціальному симетричному каналу зв'язку. Він допускає обмін на відстані до 1200 метрів із швидкістю передачі до 10 Мбіт/с між 32 вузлами (тобто 32 передавачі та приймачі) на одній лінії, використовуючи 2-провідну багатоточкову мережу. Можливе використання також 4–провідної лінії зв'язку, що робить інтерфейс *RS/TIA-485* сумісним з інтерфейсом *RS/TIA-422*. При необхідності можливе збільшення кількості вузлів ще на 31 одиницю за рахунок використаного лінійного підсилювача, який має назву повторювач (*repeater*).

Сигнал передається диференційною напругою між лініями А і В та має такі параметри:

- від –1,5 до –6 В по відношенню до лінії В для двійкової одиниці (стан «МІТКА» або «ВМИКНЕНО»);
- від +1,5 до +6 В по відношенню до лінії В для двійкового нуля (стан «ПАУЗА» або «ВИМКНЕНО»).

Лінійний формувач забезпечує по двох дротах диференціальну напругу ±5 В. Межа невизначеності приймачів складає ±200 мВ.

Головна перевага інтерфейсу це можливість формувача знаходитись у трьох станах, тобто наявність так званого *тристабільного режиму* роботи: логічна одиниця, логічний нуль та високий імпеданс. В останньому стані формувач не вживає струм і вважається, що його немає в лінії. Цей так званий стан «відключений від лінії» може вмикатись за допомогою управляючої лінії всередині мікросхеми *UART*, що допускає багатоточкове підключення вузлів з одним активним передавачем. Кожен вузол має унікальну адресу для виключення колізій. Такий інтерфейс зручний для підключення декількох датчиків або інших пристроїв або навіть ПЛК до одного ПЛК. Функцію головного пристрою може виконувати звичайний ПК, але з додатковим обладнанням – перетворювачем інтерфейсів (ПІ) *RS-232/RS-485*. Питання використання ПІ буде розглянуто пізніше. В мережі обов'язково призначається один головний пристрій, а інші будуть підпорядкованими. У такому випадку підпорядкований вузол відповідає лише на свій запит.

Насамкінець відзначимо, інтерфейс *RS-485*, описаний у стандарті *EIA/TIA-485*, є найбільш поширеним в промисловій автоматизації. Пов'язано це з тим, що за всіма основними параметрами цей інтерфейс є якнайкращим із всіх можливих. На цьому інтерфейсі побудовано більшість відомих промислових протоколів (*ModBus*) та мереж (*ProfiBus*). Основними його перевагами є:

- обмін даними здійснюється лише по одній скрученій парі дротів;
- можливість організації мережі (топологія *шина*);
- багатоточковий режим зв'язку;
- велика довжина лінії зв'язку;
- можливість живлення пристроїв;
- достатньо висока швидкість передачі даних без зниження їх якості та надійності.

В основі побудови інтерфейсу *RS-485* лежить диференціальний (симетричний) спосіб формування сигналу, коли напруга, відповідна рівню логічної одиниці або нуля, відлічується не від «землі», а вимірюється як різниця потенціалів між двома передавальними лініями: ***Data+*** та ***Data-***. При цьому напруга кожної лінії відносно «землі» може бути довільною, але не повинна виходити за діапазон – 7…+12 В.

Приймачі сигналу є диференціальними, тобто сприймають лише різницю між напругою на лінії ***Data+*** та ***Data-***. При різниці напруги більше 200 мВ, до +12 В вважається, що на лінії встановлено значення логічної одиниці, при напрузі менше –200 мВ, до – 7 В – логічного нуля.

Для реалізації багатоточкового режиму обміну в інтерфейсі *RS-485* використаний варіант побудови мережі, коли приймачі по відношенню до передавача є так званим вхідним імпедансом (опором), причому опір приймачів не обмежується, але це збільшує навантаження на передавач та робить систему чутливою до перешкод.

Інтерфейс *RS-485* працює таким чином. Перш за все на всіх трансиверах, крім інформаційних сигналів *RD* та *TD*, з'явився додатковий сигнал управління передаванням/прийманням. Тепер при закінченні передачі пристрій може відключити свій передавач (перевести його у високоімпедансний стан) і дати можливість іншим трансмітерам підключатися до

лінії для передачі. Іншими словами, сигнал дозволу передачі переводить трансмітер з активного стану в пасивний, тобто передавач може генерувати логічну „1”, логічний „0” або знаходитись в пасивному стані. В який час та в якому порядку передавачі вузлів будуть отримувати доступ до шини у стандарті не зазначено, це питання повинне вирішуватись на канальному рівні (організація доступу до шини). Реалізація інтерфейсу повинна забезпечити працездатність обладнання при короткочасних колізіях (коли два передавачі займають одночасно шину), що, до речі, теж не описано в стандарті.

У стандарті на інтерфейс *RS-485* не описано вимог до середовища передавання даних. Але найчастіше використовується двопровідна схема передачі даних на фізичному рівні. Це, наприклад, може бути екранована скручена пара. Передавач та приймач кожного вузла підключається до єдиної шини, тобто контакт «А» кожного передавача та контакт «А´» кожного приймача підключаються до загальної лінії «А», так само і контакти «В» та «В´» підключаються до лінії «В». Таким чином, коли передавач одного вузла передає послідовність бітів, всі приймачі інших вузлів їх приймають. Таким чином, в інтерфейсі *RS-485* реалізовано півдуплексний режим обміну, тобто система дозволяє в один момент часу вузлу передавати інформацію, а в інший приймати.

Бітова швидкість вибирається залежно від сумарної довжини лінії, характеристик кабелю, і, як правило, наводиться в документації до обладнання.

При високих швидкостях і при значних відстанях необхідно вирішити проблему відбиття сигналу на кінцях лінії. З курсу електротехніки відомо, що основою для мінімізації відбиття є використання узгоджувальних резисторів з номіналами, які відповідають хвильовому опору кабелю. Як правило, у промислових мережах використовують кабель з характеристичним імпедансом *120 Ом*, тому на обох кінцях шини між лініями «А» і «В» підключений узгоджувальний резистор R_t (термінатор) з відповідним опором. Зауважимо, що це потрібно робити лише на кінцевих вузлах шини, але на невеликих відстанях та невеликих швидкостях термі-

нальний резистор не потрібен. На рис. 2.6 показана схема електричних з'єднань пристроїв відповідно стандарту на інтерфейс *RS-485*.

Відзначимо, що стандарт на інтерфейс *RS-485* також не описує типи з'єднувачів. Історично склалось, що підключення вузлів до ліній зв'язку здійснюється за допомогою гвинтового з'єднання, іноді використовують пружинні з'єднувачі, які впровадила компанія *WAGO* (Німеччина). Також використовують з'єднувачі типу *DB-9*. В останньому випадку лінії «А» і «В» підключають відповідно до контактів № 8 та № 3. На клемних колодках деяких пристроїв лінії «А» і «В» позначають відповідно «RxD/TxD-N» та «RxD/TxD-P».

2.4.3. Порівняння характеристик стандартів на інтерфейс RS/TIA-232 та RS/TIA-485

Підсумуючи наведене в пунктах *2.4.1 та 2.4.2*, можна скласти порівняльну таблицю (див. табл. 2.3).

2.4.4. Принцип дії перетворювачів послідовних інтерфейсів

Розроблення цілої низки послідовних промислових стандартів зумовило необхідність створення пристроїв, які б забезпечили сумісність старих пристроїв з новими. Це в першу чергу перетворювачі інтерфейсів з *RS-232* в *RS-485*. Для конфігурування та налагоджування пристроїв з інтерфейсом *RS-485* можна використовувати звичайний ПК, але це незручно в умовах підприємства.

В таких випадках використовують спеціально захищені від потрапляння води та пилу програматори на базі *laptop* з встановленим спеціальним ПЗ або звичайні ноутбуки. Але на цей час склалась тенденція щодо заміщення в ноутбуках стандартного інтерфейсу *RS-232* на інтерфейс *USB* (*Universal Serial Bus* – універсальна послідовна шина). Тому цей інтерфейс виробники стали використовувати в ПЛК, операторських панелях (ОП), модулях збору даних тощо для їхнього програмування та конфігурування. В такому випадку для з'єднання пристроїв з ноутбуком потрібен ПІ з *USB* в *RS-485*.

Застосування ПІ може також забезпечити збільшення довжини лінії зв'язку, яка з'єднує два пристрої з інтерфейсом *RS-232*. Типова блок-схема ПІ показана на рис. 2.7.

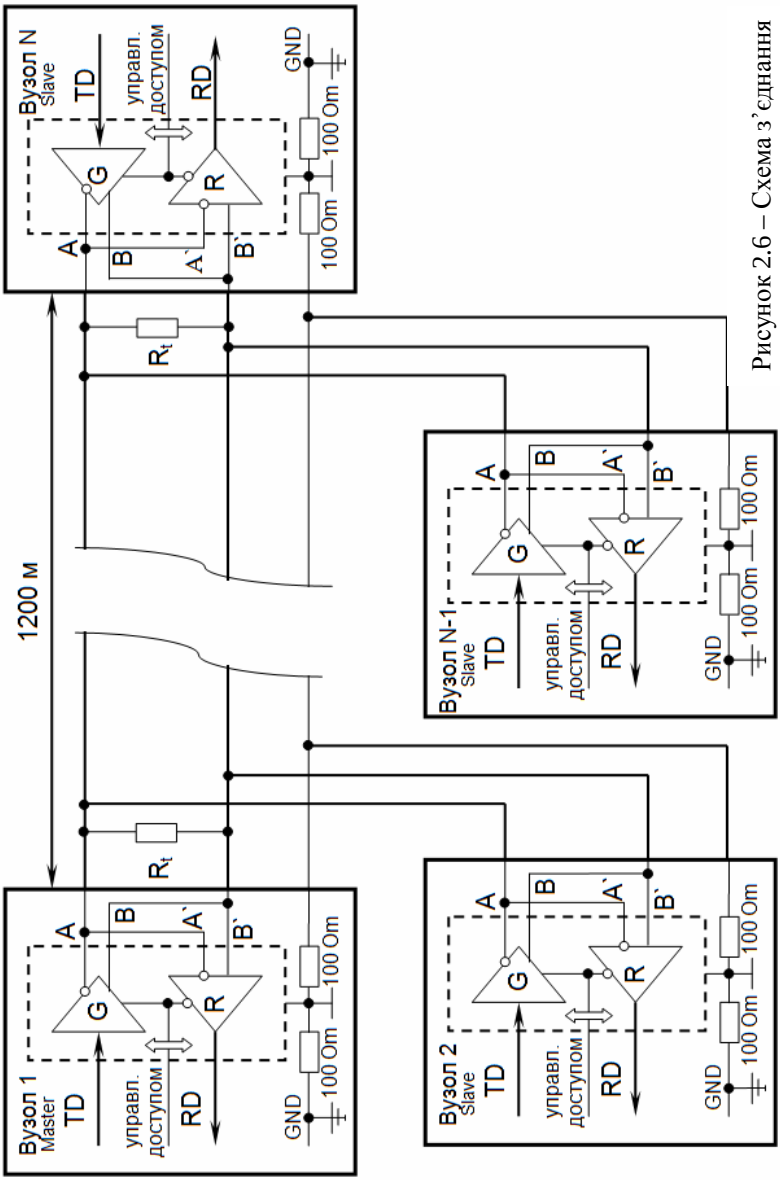


Рисунок 2.6 – Схема з'єднання вузлів інтерфейсом *RS-485*

Таблиця 2.3 – Порівняльна таблиця стандартів *RS-232* та *RS-485*

Параметр	Стандарт інтерфейсу	
	RS-232	*RS-485*
Режим роботи	Несиметричний	Симетричний (диференціальний)
Макс. кількість ПРД та ПРМ	1 ПРД, 1 ПРМ	32 ПРД, 32 ПРМ (на один сегмент мережі)
Довжина лінії зв'язку, м	15	1200
Макс. швидкість передавання даних	20 кбіт/с	10 Мбіт/с
Макс. синф. напруга, В	±25	+12 до –7
Вих. сигнал ПРД, В мін. макс.	 ±5 ±25	 ±1,5 ±6В
Вих. опір ПРД, Ом	>3 Ом	60 Ом
Вхід. опір ПРМ, Ом	3-7 кОм	>12 кОм
Чутливість ПРМ, В	±3В	±200мВ

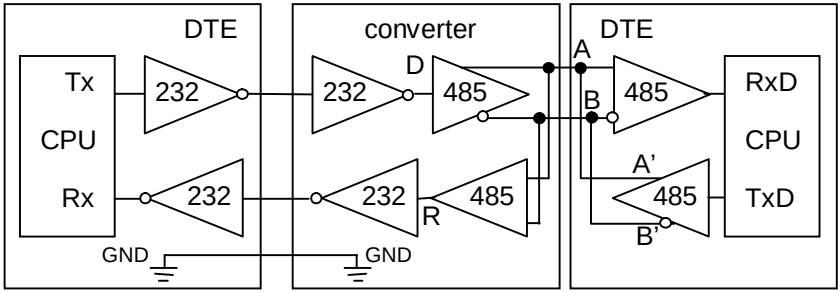


Рисунок 2.7 – Типова блок-схема ПІ *RS-232/RS-485*

Типовими параметрами ПІ типу *RS-232/RS-485* є такі:

- швидкість передачі даних до 1 Мбіт/с;
- перемичний вибір пристроїв *DCE/DTE*;

- перетворення всіх сигналів даних і управління;
- наявність світлодіодів для відображення статусу пристроїв і сигналів;
- живлення від джерела змінного струму;
- електричне розділення інтерфейсів перетворювача;
- використання різноманітних з'єднувачів з боку *RS-232* та гвинтових кріплень з боку *RS-485*.

Для забезпечення сумісності стандартів виробники пропонують різні конвертери інтерфейсів: *RS-232/RS-485*, *USB/RS-485* тощо. Оскільки ПІ працюють на фізичному рівні моделі *OSI*, то потрібно зупинитися на деталях апаратної сумісності. Крім того, існує ще одна проблема – це порядок доступу до лінії та різний тип з'єднань інтерфейсів. Передавачі *RS-485* повинні підтримувати можливість відключення від лінії зв'язку, якщо інший вузол робить передачу. Таким чином в ПІ повинно бути забезпечено переключення передавача з боку *RS-232* в активний або пасивний стан. Оскільки в інтерфейсі *RS-232* таке не передбачено, виробники конвертерів використовують різні способи. Це використання управляючих ліній *RTS* та *DTR* інтерфейсу *RS-232* для управління потоком даних в ПІ. Також можлива автоматична робота ПІ, яка використовує сигнали даних з приймального буфера. Можливе управління потоком даних з боку інтерфейсу *RS-485*, але це працює лише в режимі «точка-точка».

На рис. 2.8 показана типова схема з'єднання пристроїв *DTE* та *DCE* з різними інтерфейсами.

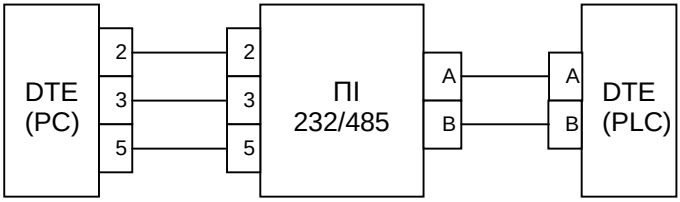


Рисунок 2.8 – Типова блок-схема ПІ *RS-232/RS-485*

У промислових мережах, поряд іззастосуванням *RS-232* та *RS-485*, поширюється застосування шини *USB* в ПЛК та ОП. Проте це поки лише

можливе для обміну конфігурування ресурсів ПЛК або ОП. Це півдуплексна, синхронна комунікаційна шина з чіткою ієрархією головний/підпорядкований пристрій з тривалістю пакета 1 мс. Застосовується код *NRZi* (див. рис. 1.12).

2.5. Практична реалізація обміну даними через послідовні інтерфейси

Для закріплення теоретичних знань про побудову та принципи використання послідовних інтерфейсів в даному розділі запропоновано виконати комп'ютерний практикум. Сутність практикуму полягає в отриманні практичних навичок налаштування апаратно-програмних засобів, за допомогою яких реалізовано обмін по послідовних інтерфейсах *RS-232* та *RS-485*.

2.5.1. Взаємодія двох комп'ютерів через інтерфейс RS-232

Отже, для реалізації обміну по інтерфейсу *RS-232* використовується два комп'ютери, які з'єднані «нульмодемним» кабелем. Можливе застосування як повного, так і мінімального «нульмодемного» кабелю. Схему з'єднання комп'ютерів наведено на рис. 2.2, а схему «нульмодемного» кабелю – на рис. 2.9.

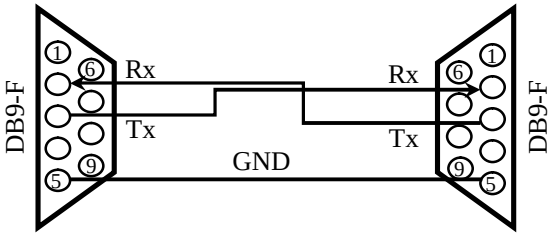


Рисунок 2.9 – Схема «нульмодемного» кабелю

У табл. 2.4 наведено варіанти налагоджувань з'єднань, попарно з'єднаних ПК, та формат кадрів, що передаються в повідомленні. У вікні налагоджування інтерфейсу *COM*-портів для всіх пар інтерфейсів *RS-232* встановіть однакові швидкість та формат кадру, що передається.

У табл. 2.5 вказано зміст повідомлень в *ASCII*- та в *HEX*-коді, якими будуть обмінюватись ПК. Таблиця з *ASCII*-кодами символів наведена в [6].

Таблиця 2.4 – Параметри налагоджування інтерфейсу *COM*-портів ПК

Параметри інтерфейсу	Робочі місця (ПК)				
	S1–S2	*S3–S4*	*S5–S6*	*S7–S8*	*S9–S10*
Швидкість, біт/с	9600	9600	9600	9600	9600
Кількість бітів у кадрі	8	8	8	7	7
Паритет	немає	пар.	непар.	пар.	непар.
Кількість стоп-бітів	1	1	1	2	2

Для реалізації обміну даними застосовується програма *HyperTerminal*, яка вбудована у стандартний набір комунікаційних утиліт ОС *Windows XP/2000/ME*. Також для сканування *COM*-портів і відправлення повідомлень використовується прикладна програма *COM Port Toolkit*, яка є безкоштовною і доступна для скачування з сайту HTTP://www.compt.ru.

Таблиця 2.5 – Зміст повідомлень, що надсилають ПК

Формат	Робочі місця (ПК)				
	S1↔S2	*S3↔S4*	*S5↔S6*	*S7↔S8*	*S9↔S10*
ASCII	(	)	:	+	,
HEX	28	29	2A	2B	2C

Після підключення кабелю до портів та вмикання комп'ютерів необхідно запустити програму *HyperTerminal.* Далі, за допомогою меню програми, потрібно створити нове підключення (на рис. 2.10 – це підключення з назвою «*COM10»)* за вказаними в табл. 2.5 параметрами інтерфейсу до *COM*-порту (наприклад, це порт № 10). Далі, після здійснення підключення по черзі передавайте повідомлення від одного ПК іншому. При цьому за умовчанням на ПК-відправнику в рядку введення повідомлення не відображатиметься. Щоб бачити результат введення символів, необхідно виставити позначку в основних властивостях з'єднання, через вкладку *Setting* в налаштуваннях символьного обміну, як це показано на рис. 2.10.

Окрім символьних повідомлень, через *HyperTerminal* можна відправляти та отримувати файли. Для цього в меню швидкого доступу *Transfer* в *HyperTerminal* є команди *Send File* і *Receive File.*

Для відправлення файлу створіть в текстовому редакторі «*Блокнот*» файл з коротким повідомленням власного змісту. Далі відправте свій файл на ПК-абонент, тобто спочатку від одного ПК на інший, потім інший файл – назад.

Окрім утілити *HyperTerminal*, існують більш зручні програми, які є прикладними та пропонуються різними розробниками програмного забезпечення. Наприклад, для роботи з *COM*-портом є раніше згадана програма *COM Port Toolkit*, яка вільно розповсюджується, але має обмеження у використанні.

Зауважимо, що з одним *COM*-портом не можуть одночасно працювати дві і більше програми (за рідкісним виключенням – це програма *ComRead*, але потрібно використати спеціальний кабель та з'єднувач-розгалужувач). Тому перед запуском програми *COM Port Toolkit* обов'язково закрийте всі програми, які мали доступ до *COM*-порту. В даному випадку це стосується утілити *HyperTerminal*. Отже, після виконання завдання відключіться від *COM*-порту та закрийте програму *HyperTerminal*.

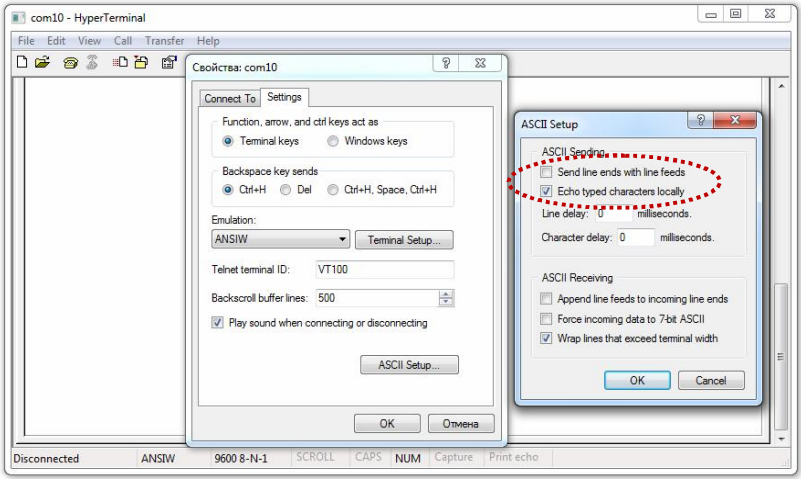


Рисунок 2.10 – Вікно налагоджування програми *HyperTerminal*

Використовуючи меню швидкого запуску, знайдіть та запустіть програму *COM Port Toolkit* на обох ПК, які з'єднані «нульмодемним» кабелем. Далі створіть та побудуйте з'єднання (профіль) з *COM*-портом. Для свого профілю відповідно до таблиці створіть повідомлення у вбудованому в програму блокноті. По черзі пересилайте повідомлення від одного ПК до іншого та спостерігайте результати пересилання в інформаційному полі. Це будуть повідомлення в *HEX*- або *ASCII*- форматі та час приходу повідомлення.

Для отримання порівняльних оцінок часу передачі повідомлень в різних форматах (*HEX* та *ASCII*) необхідно спочатку відправити п'ять повідомлень від одного ПК до іншого ПК. При цьому потрібно зафіксувати час передачі повідомлення в мілісекундах. Потім необхідно відправити інше повідомлення назад, тобто від другого ПК до першого і теж зафіксувати час передачі. При однакових налагодженнях *COM*-портів та однаковій кількості символів середній час прямої і зворотної передачі повинен збігатися.

Розглянемо передачу повідомлення, наприклад передачу *ASCII*-символу «**=**». Якщо цей символ передавати в режимі *HEX*, то програма вкаже на неможливість передачі подібного символу, оскільки у *HEX*-форматі допустимими є лише 16 символів: цифри від *0* до *9* та букви латинського алфавіту *A, B, C, D, E і F*. Проте символ «**=**» можна передати у *HEX*-форматі, використовуючи його *ASCII*-код. У цьому випадку це буде посилка двох *HEX*-символів «*3 D*». Проте, якщо передати символи «*3 D*» в рядковому форматі, то на приймальній стороні буде отримана комбінація «33 44». Крім того, відзначимо, що передача *HEX*-символів у *HEX*-форматі можлива тільки, якщо передається два символи. Розглянутий вище приклад передачі символу «**=**» вказує на необхідність визначення умов передачі даних, тобто режиму та формату передачі даних. Також необхідно враховувати, що в *ASCII*-режимі використовуються службові символи «*CR*» (повернення каретки) та «*LF*» (переклад рядка). У свою чергу, цим символам відповідають такі *HEX*-коди*: «0D»* і «*0A*», що відповідає двійковому коду: «0001101» і «0001010» або десятковому коду – «13» і «10».

Отже, після отримання середніх оцінок часу передачі побудуйте діаграму стану лінії зв'язку при відправці повідомлення в масштабі 10 мм = 1/9600. Далі, на приймальній стороні змініть швидкість передачі даних на 19 200 біт/с та повторіть дії. Спостерігайте час приходу повідомлення і прийняті символи. Порівняйте отримані результати. Вочевидь, що однакове налагодження швидкості передавання даних та параметрів кадру з обох боків лінії зв'язку є дуже важливими чинниками, про які говорилось у першому розділі.

2.5.2. Взаємодія пристроїв через інтерфейс RS-485

Далі наведені основні моменти, на які необхідно звернути увагу при роботі з портами інтерфейсу *RS-485* на прикладі використання контролера *ПЛК150* виробництва компанії «ВО ОВЕН» (http://www.owen.ua).

Для написання програми користувача необхідно мати як мінімум загальне уявлення про принципи роботи контролера з *COM*-портом. Перш ніж ПЛК почне приймати або посилати пакети необхідно відкрити та налаштувати потрібний порт. Вибір та налагодження параметрів порту визначається апаратною структурою ПЛК. Виходячи з того, що ПЛК *ОВЕН* не мають операційної системи, а діють під управлінням системи виконання, адреса портів в контролері позначається через його вказівник. У документації на ПЛК можна отримати посилання на порти його інтерфейсів. Так, в *ПЛК150* є два послідовних порти: *RS-232Debug* та *RS-485*. Ці порти мають відповідні адреси: «4» та «0». Програмування контролерів ОВЕН здійснюється в середовищі *CoDeSys V2*, яке можна безкоштовно завантажити з сайту http://www.owen.ua .

Безпосередня робота з портом – відкриття, закриття, налагодження, читання та запис реалізуються через бібліотеку *SysLibCom.lib*. Після відкриття та налаштування порту контролер автоматично починає слухати інтерфейс і збирати інформацію у вхідний буфер. Розмір буфера становить ~1.5 Кб, тому якщо контролер довгий час не прочитував інформацію і при цьому порт був відкритий, необхідно періодично обнуляти вхідний буфер. Зробити це можна, викликаючи функцію читання порту *SysComRead*, до тих пір, поки ця функція не повертатиме нульове значення. Після цього можна починати роботу з пристроями.

Насамперед контролер повинен сформувати необхідну команду, яку потрібно послати в порт. Після формування команди її можна посилати в порт. Зробити це можна, використовуючи функцію *SysComWrite*. На вході цієї функції подається масив байт та кількість необхідних для передачі байтів. Після успішної відправки функція поверне кількість відправлених байтів. Після відправки команди контролер необхідно перевести в режим читання порту. Відповідь від пристрою може прийти не відразу, а з деякою затримкою. Пов'язано це з двома речами: по-перше, пристрою необхідно отримати команду, обробити та підготувати відповідь, на це, як правило, витрачається незначний час; по-друге, в більшості пристроїв організовано затримки відповіді, для реалізації таймаутів. Якщо в пристрої правильно налаштовані мережні параметри і лінія зв'язку захищена від перешкод, то відповідь від пристрою прийде з невеликою затримкою, як правило, це не більше 50 мс. Якщо лінія зв'язку погана або періодично бувають великі перешкоди, які порушують обмін, то необхідно ввести таймер очікування відповіді, щоб контролер не чекав відповідь до безкінечності, а послав новий запит. Якщо перешкоди проходять часто, то можна організувати декілька повторних запитів однієї і тієї ж команди.

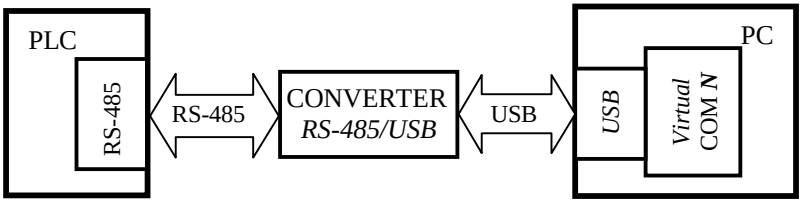
Після посилання команди необхідно перевести ПЛК в режим читання порту, для прийому відповіді. Робиться це функцією *SysComRead*. Під час надходження даних у вхідний буфер, ця функція їх звідти бере і передає у вхідний масив. При цьому функція повертає кількість прийнятих байтів. Інформацію, що приймається, можна аналізувати відразу або збирати в більший буфер (тобто склеювати кадр), і далі обробляти повністю прийняту відповідь.

Оскільки ПК не мають вбудованого інтерфейсу *RS-485*, то для зв'язку з ПЛК застосовуються ПІ *RS-232/RS-485* або *USB/RS-485*. В останньому випадку в ОС *Windows* за допомогою драйвера емулюється віртуальний *COM*-порт, з яким можуть без обмежень працювати утиліта *HyperTerminal* або програма *COM Port Toolkit*.

Номер та параметри віртуального *COM*-порту визначаються за допомогою команди «*Управление*», що викликається через контекстне меню із списку з ім'ям «*Компьютер*»

Схеми з'єднання ПК та ПЛК за допомогою автоматичних ПІ *RS-232/RS-485* та *USB/RS-485* наведено відповідно на рис. 2.11 *а* та *б*. Ці перетворювачі насамперед здійснюють функцію узгодження електричних параметрів вказаних інтерфейсів. Крім того, вони забезпечують узгодження роботи передавачів та приймачів під час обміну даними.

а)



б)

Рисунок 2.11 – Схеми з'єднання ПЛК та ПК за допомогою ПІ

У табл. 2.6 наведено варіанти налаштувань з'єднань ПЛК і ПК та формат кадрів, що передаються. У вікні налаштування інтерфейсу *COM*-портів для всіх пар інтерфейсів *RS-232* (ПК) та *RS-485* (ПЛК) встановіть однакову швидкість та формат кадру. У табл. 2.7 запропоновано зміст повідомлень для відправлення від ПЛК до ПК в *ASCII*-коді.

Для управління послідовним портом *RS-485* і відправлення повідомлень з боку *ПЛК150* необхідно використати проект з програмою користувача, яка створена в середовищі *CoDeSys* (скорочення від *Controllers Development System*, яке розроблено компанією *3S – Smart Software Solutions GmbH* із Німеччини, https://www.codesys.com, http://www.3S-software.com) [8, 9, 10, 11] на мові *ST*. У програмі використовується системна бібліотека *SysLibCom.lib*, яка входить до середовища *CoDeSys*. У комп'ютерному практикумі пропонується використати готовий проект,

який необхідно скоректувати відповідно до завдання. Проект є навчальним і реалізує раз в одну секунду відправлення повідомлень у символьному режимі передавання *ASCII*-символів. В [12, 13] наведено основи работи в середовищі *CoDeSys* та принципи реалізації обміну даними за допомогою послідовних інтерфейсів в контролерах ОВЕН.

Таблиця 2.6 – Параметри налаштування інтерфейсу *COM*-портів у ПК та ПЛК

Параметри інтерфейсу	Робочі місця (ПК)				
	S1, S2	*S3,S4*	*S5,S6*	*S7,S8*	*S9,S10*
Швидкість, біт/с	9600	9600	9600	9600	9600
Кількість бітів у кадрі	8	8	8	7	7
Паритет	немає	пар.	непар.	пар.	непар.
Кількість стоп-бітів	1	1	1	2	2

Таблиця 2.7 – Зміст повідомлення для відправлення

ПК	Текст повідомлення	ПК	Текст повідомлення
S1	The receive buffer is full	*S6*	What is pressure
S2	Start the standby pump	*S7*	What is the flow
S3	Communication parame-ters	*S8*	What is the temperature
S4	Set output bit	*S9*	What type of connection
S5	What is the time	*S10*	What type of connection

Для прийому та аналізу повідомлень від ПЛК в ПК застосовуються утиліта *HyperTerminal* або програма *COM Port Toolkit*, які вже відомі та розглянуті раніше у пункті 2.5.1.

2.6. Введення в протоколи управління потоком даних

У попередньому підрозділі було розглянуто приклади використання послідовних інтерфейсів із застосуванням елементарних протоколів управління потоком даних, які реалізовані програмно в ПК та ПЛК. На-

справді протокол обміну вирішує значно більшу кількість завдань, ніж просте управління потоком даних. Наприклад, це такі питання:

- ініціювання початку процесу обміну;
- формування кадру та його синхронізація, визначення початку та кінця кадру;
- управління потоком даних для виключення переповнення вхідного регістру приймача;
- управління інформаційними лініями в півдуплексних каналах;
- контроль цілісності даних та виявлення помилок;
- слідкування за часом від моменту відправлення повідомлення до моменту отримання підтвердження.

Кінцевою метою простих протоколів є поліпшення умов передавання даних шляхом введення додаткових процедур, наприклад, додавання затримок між символами, які передаються, або зворотне посилання отриманих символів від одержувача джерелу. Найбільш поширеними способами управління потоком даних є протокол *XON/XOFF* та *ETX/ACK*.

Протокол управління потоком даних *XON/XOFF* оснований на передаванні двох спеціальних символів та передбачає наявність дуплексного каналу зв'язку. Зазвичай, це символи *ASCII*-коди *DC1* для *XON* та *DC3* для *XOFF*. Передавач відправляє дані, поки не отримає від приймача символ *XOFF*, передача відновлюється, якщо передавач отримує символ *XON*. Типовим прикладом є буфер звичайного принтера. Причому для рівномірності процесу передавання даних посилання символів управління потоком здійснюється не в момент повного заповнення, а тоді, коли буфер заповнений на 70 %. Відповідно символ дозволу передавання відправляється, коли пам'ять буфера заповнена на 30 %. Іноді виникає проблема, якщо в потоці даних зустрінеться будь-який символ управління. Але цей випадок можна попередити забороною відсилання джерелом повідомлення з такими символами.

Час реакції передавача на зміну стану приймача порівняно з апаратним протоколом збільшується за рахунок часу передачі символу та часу реакції програми передавача на прийнятий символ. Перевагою програмного протоколу є відсутність необхідності передачі управляючих сигналів

інтерфейсу – тобто мінімальний кабель для двостороннього обміну має лише три дроти (рис. 2.9), навіть без управляючих дротів на пристроях. Недоліком, окрім часових запізнень на наявності допоміжного буфера, є складність реалізації дуплексного режиму обміну, тому що при зворотній передачі даних повинні виділятися та оброблятися управляючі символи, що обмежує набір дозволених для передавання символів.

У протоколі *ETX/ACK* для отримання чергового байту приймач відправляє команду *ACK* (*ASCII*-код 06_{hex}). У відповідь передавач відправляє приймачу один байт або пакет певного розміру.

Для передавання файлів використовують пакетний протокол *XMODEM*. Це простий протокол передавання та очікування автоматичного запиту повторної передачі (*ARQ*). Використання сигналу повторного запиту дозволяє при наявності помилки або відсутності підтвердження правильного прийому пакета автоматично повторити відправлення пакету. Тобто передавач очікує від приймача дозвіл на передачу наступного пакета (команда *ACK*) або повтор поточного (команда *NAK*). У протоколі *XMODEM* використано поле даних фіксованої довжини (128 байтів). Контрольна сума є арифметичною сумою розміром один байт. Подальшим розвитком даного протоколу є протоколи *YMODEM* та *ZMODEM*.

2.7. Взаємодія DTE-пристроїв за допомогою GSM-модемів

Для організації зв'язку між *DTE*-пристроями за допомогою телекомунікаційного обладнання міжнародними стандартами [5] запроваджено низку термінів, які визначають функціональну належність кожного елемента системи зв'язку, а саме:

1) *ME (Mobile Equipment)* – мобільне обладнання;

2) *MS (Mobile Station)* – мобільна станція;

3) *TA (Terminal Adapter)* – кінцевий пристрій-перетворювач;

4) *DCE (Data Communication Equipment)* – кінцеве обладнання каналу зв'язку (звичайний або *GSM*-модем, комутатори, хаби та конвертери) для передавання даних.

Стосовно організації зв'язку по дротових або *GSM*-каналах за допомогою модемів, підключених до керуючого пристрою по послідовному

інтерфейсу, додатково використовують терміни для позначення кінцевих керуючих пристроїв:

1) *TE (Terminal Equipment)* – кінцеве обладнання;

2) *DTE (Data Terminal Equipment)* – кінцеве обладнання для перетворення інформації користувача (ПК, роутер, касовий апарат, термінал *POS* тощо) в дані для передавання по лінії зв'язку.

Існує також пасивне комунікаційне обладнання. Це *DSU/CSU* (*Data Service Unit / Channel Service Unit*) – пристрої, які застосовують для з'єднання ліній зв'язку, розгалуження сигналів, а також з'єднувачі для підключення до пристроїв.

Під час підключення роутера користувача до провайдера за допомогою телефонних ліній зв'язку він налаштовується в якості пристрою *DTE*, проте роутер провайдера буде пристроєм *DCE*. Різниця у тому, що пристрій *DTE* має право ініціювати сеанс зв'язку, а пристрій *DCE* лише слухає лінію.

Для роботи з *GSM*-модемами застосовують *AT*-команди (перелік команд *Hayes*). Цей набір команд був розроблений компанією *Hayes* у 1997 році для модема власної розробки «*Smartmodem 300 baud*». Цей перелік команд вміщує низку коротких текстових повідомлень, які об'єднуються для формування повних операцій. Наприклад, це набирання телефонного номера, початок з'єднання або параметри підключення.

Для розпізнавання *AT*-команд вони мають бути занотованими в специфічному вигляді. Кожна команда починається з символів *AT* або *at* (від англ. *ATtention*), які доповнені однією або декількома командами. Відправлення команд здійснюється натисканням на клавішу «*Enter*». Модем оброблює команди, якщо він знаходиться в «командному режимі» або *offline*. Інакше він не відповідає та не виконує ніякі команди.

Стандартом на перелік *AT*-команд є документ під назвою *Data Transmission Systems and Equipment – Serial Asynchronous Automatic Dialing and Control*, більш відомий як *TIA/EIA-602* [15]. В подальшому організація *ITU-T* (*International Telecommunication Union – Telecommunication sector* – сектор стандартизації електрозв'язку Міжнародного союзу електрозв'язку), випустила документ *V.250*, який вміщує весь перелік команд

TIA/EIA-602 та декілька додаткових команд. Пізніше був оприлюднений стандарт *TIA/EIA-602-A*, який мав лише посилання на *V.250* та невелику історичну довідку розвитку стандарту.

Опис *AT*-команд наведено у документі *SIM300_ATC_V1.06*, який поширюється компанією *SIMCom Wireless Solutions Co., Ltd* (Кітай) [16].

Існують три типи переліку команд:

- основний (*basic*);
- додатковий (*S-parameter*);
- розширений (*extended*).

Усі команди в кожному рядку починаються з префікса AT. Закінчення команд позначається введенням символів переведення рядка <CR> ($0D_{hex}$).

Команди з основного переліку мають формат «AT<x><n>» або «AT&<x><n>», де «<x>» – це команда та «<n>» – аргумент команди. Наприклад, для включення режиму «*ЭХО*» необхідно відправити команду «ATE1». В цьому режимі модем (*DCE*) повертає до ПК (*DTE*) отримані від нього символи.

Команди з додаткового переліку *S*-типу мають формат «ATS<n>=<m>», де «<n>» – це індекс *S*-регістру для встановлення та «<m>» – значення для присвоєння.

Команди з розширеного переліку бувають чотирьох типів:

- команда типу «*test*» для перевірки модема на його функціональність та визначення діапазону можливих значень параметрів – «AT+<x>=?»;
- команда типу «*read*» для визначення поточних налаштувань модема та значень параметрів – «AT+<x>?»;
- команда типу «*write*» для встановлення в модемі налаштувань користувача – «AT+<x>=<…>»;
- команда типу «*execution*» слугує для зчитування та виконання незмінних параметрів у модемі – «AT+<x>».

На *AT*-команди модем відповідає залежно від типу та змісту команди, а також може генерувати повідомлення про різні події: виклик, отри-

мання *SMS* тощо. У табл. 2.8 наведені відповіді та повідомлення модема в текстовому та кодовому форматі.

Таблиця 2.8 – Повідомлення та відповіді модема на *AT*-команди

Текстове повідомлення	Код повідомлення	Розшифрування повідомлення
OK	0	Готовий до виконання команд
CONNECT	1	З'єднання встановлено. *DCE* перемикається з «командного» режиму в режим передавання даних *online*
RING	2	*DCE* визначило вхідний виклик з мережі
NO CARRIER	3	З'єднання розірвано або неможливо встановити з'єднання
ERROR	4	Команда не розпізнана, перебільшена довжина рядка, недопустиме значення параметра або інші помилки в командному рядку
NO DIALTONE	6	Не визначений режим тонового виклику
BUSY	7	Сигнал режиму чергування
NO ANSWER	8	«@» використовується модифікатор виклику. Немає відповіді від абонента, який викликається протягом 5 с
CONNECT <text>	визначена виробником	Специфічна інформація для *DTE*: швидкість, наявність помилок, статус тощо

Нижче наведено приклад *AT*-команд з відповідями модему та коментарями.

AT+CCUG? *Запит поточних налаштувань модема*
+CCUG: 1, 10, 1 *Відповідь модема*
OK

AT+CCUG= ,9 *Встановлення нового значення другого параметру*
OK
AT+CCUG? *Запит поточних налаштувань модема*
+CCUG: 1, 9, 1 *Відповідь модема*
OK

Як зазначено в попередньому підрозділі, для управління потоком даних використовують два способи: програмний та апаратний. Нагадаємо, що програмне управління потоком полягає у відправці в модем двох символів для старту «*XON*» та зупинки «*XOFF*» потоку даних. Такий спосіб зручний при використанні «мінімального» кабелю. Для включення програмного управління необхідно послати команду «AT + IFC = 1, 1». Однак для того, щоб цей режим залишився активним після перезавантаження модема, необхідно подати команду «AT & W» для запису параметрів налаштування в енергонезалежну пам'ять модема.

Апаратне управління потоком даних реалізовано за допомогою управляючих сигналів *RTS/CTS*. Якщо буфер пам'яті модема заповнений, то сигнал *CTS* скидається (*FALSE*) для тимчасового припинення передавання даних.

Нижче поданий перелік деяких *AT*-команд:

1) ATE0 – вимкнути режим «*ЭХО*»;

2) ATE1 – увімкнути режим «*ЭХО*»;

3) AT+IPR=<швидкість> – встановити швидкість обміну даними для послідовного порту;

4) AT&W – записати в енергонезалежну пам'ять раніше встановлені параметри;

5) AT+CREG? – запит інформації про реєстрування в мережі *GSM*. Відповідь модема буде такого формату: «+CREG:<n>,<m>»,
де «n» може набувати таких значень:

0 – звіт про реєстрування в мережі вимкнений;

1 – звіт про реєстрування в мережі увімкнений;

«m» може набувати таких значень:

0 – не зареєстровано;

1 – зареєстровано;

2 – йде пошук мережі;

3 – в реєстрації відмовлено;

4 – зареєстровано в роумінгу;

6) ATD<номер> CONNECT<швидкість> – встановлення *CSD*-з'єднання;

7) AT+CMGF=1 – встановлення режиму передавання текстових повідомлень, тобто режиму для передавання *SMS*-повідомлень;

8) AT+CMGS=<номер> відповідь модема **>**

<текст> відповідь модема **<текст>**

Ctrl+Z ($1A_{hex}$) відповідь модема **ОК** – для відправлення *SMS*-повідомлення.

Відповідь модема починається та закінчується послідовністю символів переведення рядка та повернення каретки: <$0D_{hex}$> та <$0A_{hex}$>.

Якщо модем отримав *SMS*-повідомлення, то він генерує звіт такого змісту:

+CMTI:<пам'ять>,<індекс>,

де <пам'ять> – тип використаної пам'яті модема, де записано повідомлення;

<індекс> – порядковий номер повідомлення в пам'яті.

Прочитати прийняте повідомлення можна за допомогою команди: AT+CMGR=<індекс>,0.

Взаємодія ПК та GSM-модема для відправлення SMS-повідомлення

Після підключення модема до *COM*-порту ПК та вмикання його живлення запустіть утиліту *HyperTerminal* та налаштуйте інформаційне з'єднання ПК з модемом

```
AT                  (*перевірка зв'язку з модемом *)
OK

AT+CPIN=1111        (*введення PIN-коду, якщо необхідно*)
OK

ATE0                (*вимкнення луна в каналі RS485*)
OK
```

AT+CBST=7 (*вибір швидкості обміну 9600 біт/сек*)
OK

AT+CMEE=1 (*увімкнення виведення коду похибок*)
OK

AT+CMGF=1 (*текстовий формат повідомлення *) OK

AT+CSCS="GSM" (*встановлення алфавіту GSM для *SMS*-повідомлень*)
OK

AT&W (*запам'ятовування налаштувань*)
OK

Далі, використовуючи *AT*-команди, відправте коротке повідомлення на номер абонента. Приклад відправлення *SMS*-повідомлення наведено нижче.

AT+CMGS="**********" (*відправка *SMS*: вказати номер та натиснути Enter*)
> Hello, friend (*модем виводить символ очікування тексту «**>**», далі набрати текст повідомлення в кінці тексту ввести CTRL+z *)
+CMGS: 118
OK (*модем повідомляє про виконання команди відправлення повідомлення*)

AT+CNMI=2,1,0,0,0 (*формат повідомлення модема про отримання *SMS**)
OK

Після отримання повідомлення з телефону абонента прочитати його можна за допомогою утиліти *HyperTerminal*. Нижче показано повідомлення модема про прийняте *SMS*-повідомлення, команда на читання та команда на видалення *SMS*-повідомлення з пам'яті модема:

+CMTI: "SM",2 (*повідомлення модема про отримання СМС, яке знаходиться в 2-му регістрі пам'яті *SIM*-картки*)

AT+CMGR=2 (*читання *SMS* з 2-го регістру*)

+CMGR:"REC UNREAD","+38**********", ,"05/05/15, 17:35:54+16"
Privet, modem!
OK

AT+CMGD=2 (*очищення 2-го регістру*)
OK

AT+CMGR=2 (*читання пустого регістру*)
OK.

Контрольні запитання:

1) Перелічить міжнародні установи, яки розробляють стандарти у сфері телекомунікацій.

2) Які лінії та для чого використовують в «мінімальному» «нуль-модемном» кабелі?

3) Яким чином та за допомогою яких ліній реалізується апаратне управління передачею даних по інтерфейсу *RS-232C*?

4) Поясніть, чому при збільшенні швидкості приймання даних спотворюється посилання при однаковому форматі кадрів?

5) У чому полягає принцип програмного управління потоком даних по інтерфейсу *RS-232C*?

6) Які функції виконує мікросхема *UART* в послідовних інтерфейсах *RS-232C*?

7) Який метод кодування інформації застосовано в інтерфейсі *RS-485?*

8) Яким чином стало можливим багатоточкове з'єднання в інтерфейсі *RS-485?*

9) Які переваги інтерфейсу *RS-485* в порівнянні з інтерфейсом *RS-232* при використанні в умовах промисловості?

10) Яким чином працюють автоматичні перетворювачі інтерфейсів *RS232/RS485* та *USB/RS485*?

11) Який режим обміну реалізований в інтерфейсі *RS-485*?

12) Для чого використовують стартовий та стоповий біти при символьному обміні даними?

13) Яким чином працює алгоритм перевірки на парність?

14) Чим відрізняється принцип кодування бітів в інтерфейсах *RS-232* та *RS-485?*

15) Якого типу бувають команди з переліку *AT*-команд?

16) Чому при з'єднанні модема та ПЛК по інтерфейсу *RS-485* потрібно вимикати режим «ЛУНА»?

17) Які налаштування потрібно зробити для підключення модема до ПК за допомогою програми *HyperTerminal*?

18) Які режими обміну даними забезпечує *GSM*-модем?

19) Які повідомлення і в якому форматі може відправляти модем у випадку виникнення помилок?

РОЗДІЛ 3.
ОСНОВИ ВИКОРИСТАННЯ ПРОТОКОЛІВ ПОСЛІДОВНИХ ІНТЕРФЕЙСІВ У ПРОМИСЛОВИХ МЕРЕЖАХ

У цьому розділі розглянуто теоретичні та практичні питання використання протоколів послідовних інтерфейсів в промислових мережах. Наведено теоретичні і практичні відомості щодо налагодження програмних модулів для реалізації протоколів послідовного обміну.

Так, завершивши вивчення цього розділу, ви зможете:

- отримати уявлення про принципи формування запитів та відповідей згідно з протоколами *ModBus*, *OWEN* та *DCON*;

- ознайомитись зі складом кадрів протоколів послідовного обміну;
- налагоджувати апаратно-програмні засоби систем обміну даними, які використовують протоколи послідовних інтерфейсів.

3.1. Загальна характеристика апаратно-програмних засобів реалізації протоколів послідовних інтерфейсів

Реалізація протоколів послідовних інтерфейсів у даному розділі розглянута на основі використання контролерів 100-ї серії виробництва компанії «ВО ОВЕН» та прикладного програмного забезпечення (середовище *CoDeSys V2*). Розгляд основних принципів формування запитів згідно з протоколами *ModBus*, *OWEN* та *DCON* побудований за такою схемою. Контролер налагоджується та виконує функцію пристрою з тим або іншим протоколом у статусі ведучого або веденого, а прикладні програми, які запущені на ПК, виконують функції веденого або ведучого віртуального пристрою. Всі протоколи використовують адресну модель ідентифікації ведених пристроїв, тобто вимагають призначення адреси веденому пристрою у мережі. Зазвичай головний пристрій не має адреси, а широкомовне повідомлення відправляється за адресою з номером «0». Основними параметрами налагодження інтерфейсу є швидкість передавання даних та формат кадру, в якому вказується кількість бітів даних і стопбітів, та наявність або відсутність біта паритету. Крім того, користувач додає потрібні мережні змінні для обміну даними. В параметрах мережних змінних вказується номер регістра та назва функції, яка застосовується (для протоколу *ModBus*), *HASH* ім'я змінної (для протоколу *OWEN*) або формат запиту і відповіді (для протоколу *DCON*). Окремо вказується режим обміну даними: *ASCII* (передавання символів) або *RTU* (передавання байтів).

Каналом зв'язку між ПЛК та ПК є лінії з інтерфейсом *RS-232* або *RS-485*. Оскільки ПК не мають вбудованого інтерфейсу *RS-485*, то для зв'язку з ПЛК застосовуються ПІ *RS-232/RS-485* або *USB/RS-485*. Зауважимо, що в останньому випадку ОС за допомогою драйвера створює віртуальний *COM*-порт. Номер та параметри віртуального *COM*-порту визначаються за допомогою «*Диспетчера устройств*», який входить у ста-

ндартний набір сервісів операційних систем типу *Windows*. Схеми з'єднань ПЛК та ПК за допомогою ПІ різних типів показано на рис. 2.11, *а* та *б*.

Як допоміжний засіб для вивчення протоколів використовують прикладні програми, які дозволяють вивчити принципи побудови запитів та відповідей, а також правильно їх інтерпретувати. Для зв'язку ПК та ПЛК по послідовному інтерфейсу за допомогою ПІ *RS-485/USB* в ОС *Windows* за допомогою драйвера емулюється віртуальний *COM*-порт, з яким без обмежень можуть працювати програми *COM Port Toolkit* та *ComRead*. Проте ці програми не можуть інтерпретувати запити і формувати відповіді. Для аналізу кадрів протоколу *ModBus* необхідно використовувати спеціальні програми *CAS Modbus Scanner* та *CAS Modbus RTU Parser* (http://www.chipkin.com). Також використовуються програма-симулятор *веденого* пристрою *ModSim32* (http://www.win-tech.com) і програма-симулятор *ведучого* пристрою *OWEN Test Console* (http://owen.ua). Для реалізації протоколів *ModBus*, *OWEN* та *DCON* у контролері *ПЛК150 ОВЕН* використовується середовище програмування контролерів *CoDeSys* з набором модулів, які додаються до основної конфігурації ресурсів ПЛК. Керівництво з експлуатації та порядок конфігурування мережних модулів контролерів ОВЕН представлені в [14] та [15].

3.2. Основні відомості про протокол *Modbus*

Протокол обміну даними *ModBus* було розроблено компанією *Gould Modicon* (зараз входить до французької компанії *Schneider Electric*, www.schneider-electric.com) для контролерів власного виробництва. Вперше специфікація протоколу була опублікована в 1979 році. Стосується вона до протоколів відкритого типу. Протокол *ModBus* на сьогоднішній день є одним з найбільш популярних. Основна причина такої популярності – це відкритість та простота в реалізації. Хоча протокол *Modbus* є відносно повільним порівняно з іншими шинами, він має перевагу в тому, що широко поширений серед виробників та користувачів вимірювальних приладів. Устаткування з протоколом *Modbus* випускають від 20 до 30 виробників, а в промисловості вже давно використовуються багато подібних систем, тому цей протокол можна вважати за промисловий стандарт *de facto*.

На сьогоднішній день протокол *ModBus* підтримує і розвиває організація *MODBUS-IDA* (http://www.modbus.org), яка забезпечує його відкритість та розробляє готові компоненти для спрощення реалізації. Відповідно до стандартів *MODBUS-IDA* – *ModBus* є протоколом прикладного рівня для організації зв'язку типу «Клієнт – Сервер» між прикладними Процесами пристроїв, які під'єднані до різноманітних типів шин або мереж. В контексті моделі *OSI* ці мережі мають архітектуру, яка показана на рис. 3.1. Користувач може вибирати між інтерфейсами *RS-232, RS-422, RS-485* або струмовою петлею (*CL*) 20 мА, кожен з яких підходить для швидкостей передач даних, визначуваних протоколом. Стек протоколів *ModBus* на сьогоднішній день представлений чотирма мережами: *ModBus RTU* та *ASCII* (*ModBus over Serial Line*), *ModBus Plus* (на рис. 3.1 не показаний) і *ModBus TCP/IP*. На послідовних інтерфейсах протокол *ModBus* складається з трьох рівнів: прикладного, канального та фізичного [16]. У випадку використання каналів *Ethernet* протокол *ModBus* побудований поверх транспортного та мережного рівнів. Тому використаний додатковий рівень *ModBus on TCP/IP*, який не описаний в моделі *OSI*.

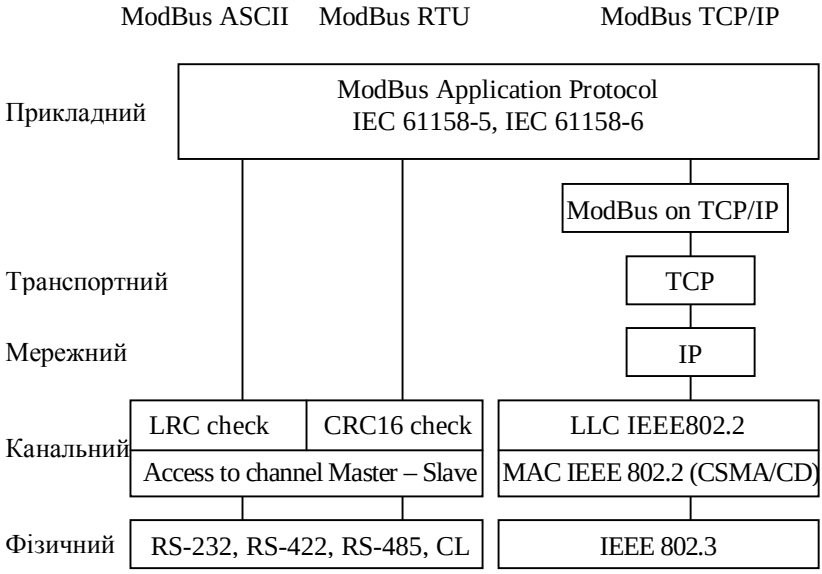


Рисунок 3.1 – Протокол *ModBus* в контексті моделі *OSI*

ModBus Application Protocol (*MBAP ModBus* – протокол прикладного рівня) базується на моделі обміну повідомленнями «Клієнт – Сервер» і визначає формат повідомлень *ModBus PDU* (*Protocol Data Unit* – простий протокол блоків даних), які мають вигляд, що показаний на рис. 3.2. Клієнтський прикладний *Процес* робить повідомлення-запит до серверного *Процесу*, в якому у полі «код функції» вказує йому на дію, яку необхідно провести. Байти даних вміщують інформацію, яка необхідна для виконання даної функції. Серверний прикладний *Процес* у випадку вдалого виконання цієї функції повторює код функції у відповіді (якщо запит передбачає відповідь). При виникненні помилки код функції у відповіді модифікується (старший біт виставляється в «1»), а в байтах даних передається причина помилки. Тобто, якщо при передачі клієнтським прикладним *Процесом* повідомлення-запиту з функцією 03_{16} (0000_0011_2) виникла помилка у її виконанні Сервером, той відішле відповідь з полем функції, рівним 83_{16} (1000_0011_2). В доповненні до зміни коду функції, при помилці, Сервер розміщує в поле даних унікальний код, який вказує на тип і причину помилки.

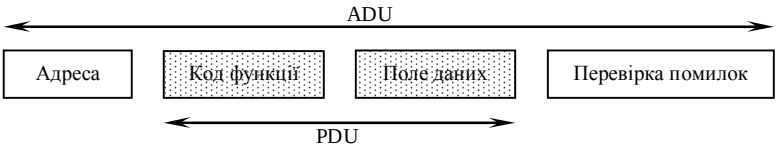


Рисунок 3.2 – Кадри прикладного та канального рівнів протоколу *ModBus*

Код функції відповідає полю розміром в один байт, яке може набувати значення від 1 до 255 (коди 128-255 зарезервовані під коди повідомлень-відповідей при помилкових діях). В табл. 3.1 показано розподіл функцій за номерами згідно з їх належністю.

Стандартні функції визначаються організацією *MODBUS-IDA*. У цьому переліку є як затверджені, так і ще невикористані функції. Функції користувача визначаються виробником обладнання або користувачем. Але при цьому не гарантується, що інший пристрій не буде застосовувати цей же код для виконання іншої функції. Коди функцій, які зарезервовані, про-

те вже використані різними виробниками, мають наступні номера: 9, 10, 13, 14, 41, 42, 90, 91, 125, 126 та 127.

Таблиця 3.1 – Розподіл функцій у протоколі *ModBus*

Коди функцій	Належність функцій
1–64	Стандартні
65–72	Користувача
73–99	Стандартні
100–110	Користувача
111–127	Стандартні

Розглянемо деякі функції, які призначені для доступу до даних процесу. Ці дані, з точки зору *ModBus*-функцій, поділяються на:

- *Discrete Inputs*: дискретні входи, доступні лише для читання;
- *Coils*: котушки, внутрішні біти або дискретні виходи, для читання та запису;
- *Input Registers*: вхідні 16-бітні змінні, доступні лише для читання;
- *Holding Registers*: внутрішні/вихідні 16-бітні змінні, для читання та запису.

У повідомленні-запиті за полем коду функції можуть слідувати дані, які уточнюють або доповнюють функцію допоміжними даними. Це можуть бути адреси змінних, їх кількість, лічильник байтів даних та самі дані для запису. Для певних функцій поле даних може бути відсутнім взагалі. Максимальна довжина повідомлення прикладного рівня дорівнює 253 байти.

Повний список кодів функцій є в специфікації протоколу [16]. Номер функції дається в *HEX*-форматі. Скорочення в дужках *Hi* та *Lo* вказують відповідно на старший та молодший байти. Тобто, якщо для вказівки адреси початкової змінної необхідно двохбайтове слово, то значення старшого байта буде передаватись в полі з позначенням *Hi*, а молодшого – відповідно *Lo*.

Код функції 01_{16} – це читання статусу дискретних вихідних бітів. Повідомлення-запит вміщує адресу початкового біта і кількість бітів для читання. Біти нумеруються, починаючи з адреси «0». У повідомленні-

відповіді кожне значення змінної передається одним бітом, тобто в одному байті пакується статус 8 бітових змінних. Якщо кількість їх не кратна восьми, інші біти в байті заповнюються нулями. Лічильник вміщує кількість байтів в полі даних. В табл. 3.2 наведена структура запиту та відповіді для функції з кодом 01_{16}.

Код функції 02_{16} – це читання статусу дискретних входів. Формат даного запиту такий же як попереднього, за винятком поля функції.

Таблиця 3.2 – Структура запиту та відповіді для функції з кодом 01_{16}

ЗАПИТ	
Код функції	01
Адреса початкового біта (*Hi*)	0…$FFFF_{16}$
Адреса початкового біта (*Lo*)	
Кількість бітів (*Hi*)	1…$7D0_{16}$ (2000)
Кількість бітів (*Lo*)	

ВІДПОВІДЬ	
Код функції	01
Лічильник байтів	N
Значення бітів (перші 8 біт)	0…FF_{16}
Значення бітів (наступні 8 біт)	0…FF_{16}
…	
Значення бітів (N-ні 8 біт)	0…FF_{16}

Код функції 03_{16} – це читання значення вихідних/внутрішніх регістрів. Повідомлення-запит вміщує адресу початкового вихідного/внутрішнього регістру (слово розміром два байти) і кількість регістрів для читання. Регістри нумеруються, починаючи з адреси «0». У відповідному повідомленні в полі даних кожний регістр передається двома байтами. В табл. 3.3 наведена структура запиту та відповіді для функції з кодом 03_{16}.

Код функції 04_{16} – це читання значення вхідних регістрів. Формат даного запиту такий же як попереднього, за винятком поля функції.

Код функції 05_{16} – це запис вихідного/внутрішнього біта. В запиті вказується номер бітової змінної та значення: «0» – 0000_{16}, а «1» – $FF00_{16}$, всі інші значення не міняють стану змінних. В широкомовній передачі клієнтський запит виставляє значення даної змінної для всіх серверів.

Нормальна відповідь серверу є повторенням запиту до клієнта. В табл. 3.4 показано структура запиту на встановлення або скидання одного біта для функції з кодом 05_{16}.

Таблиця 3.3 – Структура запиту та відповіді для функції з кодом 03_{16}

ЗАПИТ	
Код функції	03
Адреса початкового регістру (*Hi*)	0…$FFFF_{16}$
Адреса початкового регістру (*Lo*)	
Кількість регістрів (*Hi*)	1…$7D_{16}$ (125)
Кількість регістрів (*Lo*)	

ВІДПОВІДЬ	
Код функції	03
Лічильник байтів	N*2
Значення 1-го регістру (*Hi*)	0…$FFFF_{16}$
Значення 1-го регістру (*Lo*)	
…	
Значення N-го регістру (*Hi*)	0…$FFFF_{16}$
Значення N-го регістру (*Lo*)	

Таблиця 3.4 – Структура запиту для функції з кодом 05_{16}

ЗАПИТ	
Код функції	05
Адреса біта (*Hi*)	0…$FFFF_{16}$
Адреса біта (*Lo*)	
Значення біта (*Hi*)	0000_{16} або $FF00_{16}$
Значення біта (*Lo*)	

Код функції 06_{16} – це запис вихідного/внутрішнього регістру. Функція аналогічна попередній, але оперує з регістрами (словами). В запиті вказується номер вихідного/внутрішнього регістру та його значення. В широкомовній передачі запит виставляє значення даної змінної для всіх серверів. Нормальна відповідь сервера є повторенням запиту клієнту. В табл. 3.5 наведено структура запиту на запис значення до одного регістру для функції з кодом 06_{16}.

Таблиця 3.5 – Структура запиту для функції з кодом 06_{16}

ЗАПИТ	
Код функції	06
Адреса регістру (*Hi*)	0…$FFFF_{16}$
Адреса регістру (*Lo*)	
Значення регістру (*Hi*)	0000_{16} … $FFFF_{16}$
Значення регістру (*Lo*)	

Код функції $0F_{16}$ – це запис декількох вихідних/внутрішніх бітів. В запиті вказується початкова адреса біта, кількість бітів для запису, лічильник байтів і безпосередньо значення. В широкомовній передачі біти записуються всім серверам. Розглянемо приклад для встановлення наступних бітових вихідних/внутрішніх змінних. Потрібно записати значення в бітах, починаючи з біта № 44 по біт № 56 згідно з табл. 3.6.

Таблиця 3.6 – Значення, які потрібно записати в біти з №44 до №56

Байт 1								Байт 2							
51	50	49	48	47	46	45	44	-	-	-	56	55	54	53	52
1	1	0	1	1	0	0	1				0	1	1	0	0

У табл. 3.6 показана відповідність адреси змінної, починаючи з 44-ї, і значення біта, усього потрібно записати значення до 13 змінних типу «біт» (56-44=12 та включаючи біт № 44 буде всього 13 бітів). Зауважимо, що адреса початкового біта має значення «2B», яке розраховується шляхом віднімання одиниці від номера біта та переведення його у *HEX*-форматі. Для зручності біти розміщені у тому порядку, як і передаються. В другому байті корисні лише п’ять перших бітів, значення інших не береться до уваги, оскільки кількість бітів вказана у кадрі. Запит та відповідь будуть відповідати табл. 3.7.

Код функції 10_{16} – запис декількох вихідних/внутрішніх регістрів. У запиті вказується початкова адреса регістру, кількість регістрів для запису, лічильник байтів і безпосередньо значення. В широкомовній передачі регістри записуються всім серверам. В табл. 3.8 наведена структура запиту та відповіді для функції з кодом 10_{16}.

Таблиця 3.7 – Структура запиту та відповіді для функції з кодом $0F_{16}$

ЗАПИТ	
Код функції	0F
Адреса початкового біта (*Hi*)	00
Адреса початкового біта (*Lo*)	2B
Кількість бітів (*Hi*)	00
Кількість бітів (*Lo*)	0D
Лічильник байтів	02
Дані (змінні 44–51)	D9
Дані (змінні 52–56)	0C

ВІДПОВІДЬ	
Код функції	0F
Адреса початкового біта (*Hi*)	00
Адреса початкового біта (*Lo*)	2B
Кількість бітів (*Hi*)	00
Кількість бітів (*Lo*)	0D

Таблиця 3.8 – Структура запиту та відповіді для функції з кодом 10_{16}

ЗАПИТ	
Код функції	10
Адреса початкового регістру (*Hi*)	0 … $FFFF_{16}$
Адреса початкового регістру (*Lo*)	
Кількість регістрів (*Hi*)	1 … $007B_{16}$ (123)
Кількість регістрів (*Lo*)	
Лічильник байтів	2*N
Дані (1-й регістр *Hi*)	0 … $FFFF_{16}$
Дані (1-й регістр *Lo*)	
…	
Дані (N-й регістр *Hi*)	0 … $FFFF_{16}$
Дані (N-й регістр *Lo*)	

ВІДПОВІДЬ	
Код функції	10
Адреса початкового регістру (*Hi*)	00
Адреса початкового регістру (*Lo*)	2C
Кількість регістрів (*Hi*)	00
Кількість регістрів бітів (*Lo*)	0D

Розглянемо приклад для запису значення до вихідних/внутрішніх регістрів змінних. Потрібно записати значення в змінні, починаючи з регістра № 74 по регістр № 77 згідно з табл. 3.9. Як і для бітів адреса регістру розраховується шляхом віднімання одиниці та перетворення в *HEX*-формат. Кількість байтів визначається як подвійна кількість регістрів.

Таблиця 3.9 – Значення, які потрібно записати в регістри з № 74 до № 77

	Регістр № 74	Регістр № 75	Регістр № 76	Регістр № 77
Адреса регістру (HEX)	49	4A	4B	4C
Значення регістру (DEC)	32767	65535	1000	255
Значення регістру (HEX)	7FFF	FFFF	03E8	00FF

В табл. 3.10 наведена структура запиту та відповіді для функції з кодом 10_{16}.

Таблиця 3.10 – Структура запиту та відповіді для функції з кодом 10_{16}

ЗАПИТ	
Код функції	10
Адреса початкового регістру (*Hi*)	00
Адреса початкового регістру (*Lo*)	49
Кількість регістрів (*Hi*)	00
Кількість регістрів (*Lo*)	04
Лічильник байтів	8
Дані (регістр № 74 *Hi*)	7F
Дані (регістр № 74 *Lo*)	FF
Дані (регістр № 75 *Hi*)	FF
Дані (регістр № 75 *Lo*)	FF
Дані (регістр № 76 *Hi*)	03
Дані (регістр № 76 *Lo*)	E8
Дані (регістр № 77 *Hi*)	00
Дані (регістр № 77 *Lo*)	FF

ВІДПОВІДЬ	
Код функції	10
Адреса початкового регістру (*Hi*)	00
Адреса початкового регістру (*Lo*)	49
Кількість регістрів (*Hi*)	00
Кількість регістрів (*Lo*)	08

Якщо запит не може бути виконаним або є інші помилки, то ведений пристрій, окрім встановлення біта помилки в коді функції, відправ-

ляє ведучому пристрою код помилки в полі даних. Стандартні коди помилок наведено у переліку:

- 01 – прийнятий код функції не може бути оброблений;
- 02 – адреса даних, яка вказана в запиті, не доступна;
- 03 – величина, яка міститься в полі даних запиту, є неприпустимою величиною;
- 04 – поки підпорядкований намагався виконати дію, виникла помилка, яка не відновлюється;
- 05 – підпорядкований прийняв запит та обробляє його, але це вимагає більше часу. Ця відповідь попереджує головний пристрій від генерації помилки тайм-ауту;
- 06 – підпорядкований зайнятий обробкою команди. Головний повинен повторити повідомлення пізніше, коли підпорядкований звільниться;
- 07 – підпорядкований не може виконати програмну функцію, прийняту в запиті. Цей код повертається для невдалого програмного запиту, що використовує функції з номерами 13 або 14. Головний пристрій повинен запитати діагностичну інформацію або інформацію про помилки від підпорядкованого.
- 08 – підпорядкований намагається читати розширену пам'ять, але виявив помилку паритету. Головний може повторити запит, але зазвичай в таких випадках потрібний ремонт.

3.2.1. Реалізація протоколу ModBus на послідовних інтерфейсах

Перші мережі *ModBus* базувалися на асинхронних послідовних лініях зв'язку та отримали назву *ModBus RTU* та *ModBus ASCII*. На фізичному рівні вони використовують стандартні послідовні інтерфейси з символьним режимом передачі. На сьогоднішній день в *MODBUS-IDA* ці мережі отримали назву *ModBus over Serial Line* й описані у відповідному стандарті [16]. У ньому вказуються правила та рекомендації використання на канальному та фізичному рівні. Оскільки мережа *ModBus RTU/ASCII* має шинну топологію, то визначений метод доступу до шини − *Ведучий/Ведений*. У послідовних мережах *Процес Ведучого* пристрою завжди є *Клієнтом*, а *Процеси Ведених* пристроїв – *Серверами.* Це значить, що

Ведучий відсилає запити, а *Ведені* їх обробляють. Цей запит може бути адресований як індивідуальному вузлу, так і усім *Веденим* на шині (запит типу *broadcast* – широкомовний запит).

На канальному рівні *ModBus RTU/ASCII* використовується адресація, орієнтована на ідентифікатори вузлів. Кожний *Ведений* повинен мати свою унікальну адресу (від 1 до 247), *Ведучий* не адресується, тобто не має адреси. При індивідуальних запитах *Ведучий* з клієнтським *Процесом* формує кадр із повідомленням-запитом і відправляє його за вказаною адресою. *Ведений* з серверним *Процесом* отримує цей кадр та обробляє повідомлення. Після його обробки *Ведений* формує кадр з повідомленням-відповіддю і відправляє його назад *Ведучому*. Кадр з повідомленням-відповіддю носить також функції кадру підтвердження, який *Ведучий* буде чекати від *Веденого* протягом часу, визначеного тайм-аутом.

При широкомовних запитах (*broadcast*) використовується нульова адреса. Широкомовні запити не потребують підтвердження, тому після відправки широкомовного кадру *Ведучий* не очікує кадр відповіді.

На рис. 3.2 показано загальний вигляд кадру *ModBus Serial*. Зверніть увагу, що розмежування між кадрами та тип контрольної суми тут не вказані, оскільки це залежить від режиму передачі *ASCII* або *RTU*. В полі адреси пристрою *Ведучий* при запиті вказує адресу отримувача, а *Ведений* при відповіді – свою адресу. Поля *ModBus PDU* описані вище (див. рис. 3.2).

У режимі *ModBus ASCII* кадри передаються безперервним потоком з паузами до 1с. Розділяють кадри передаванням *ASCII*-символу «:» на початку кадру. Цей режим належить до символьної передачі даних і застосовується для реалізації діагностичних функцій.

Режим *ModBus RTU* передбачає використання 8 біт даних в 11-бітному символі, що дозволяє передавати по байту на символ. Формат символу в *RTU* режимі: 1 стартовий біт; 8 біт даних (молодший біт передається першим); 1 біт паритету + 1 стоповий біт або без паритету + 2 стопових біти. Цей режим у два рази швидше, ніж режим символьної передачі.

Формат кадру *ModBus RTU* наведений на рис. 3.3. Розмежування між кадрами проводиться за допомогою часових пауз між символами.

Новий кадр не повинен з'являтися на шині раніше, ніж через 3.5*τ, від попереднього, де τ – час передачі одного символу (величина, яка зворотна швидкості передачі). Якщо час відсутності сигналу на лінії (інтервал тиші) буде більше ніж 3.5*τ, приймач ідентифікує помилку. З іншого боку, поява нового кадру раніше ніж 3.5*τ теж приведе до помилки.

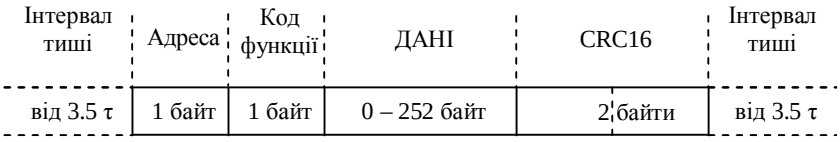


Рисунок 3.3 – Формат кадру протоколу *ModBus RTU*

Поля адреси і коду функції в режимі *RTU* займають по одному байту. Як контрольна сума використовується два байти, обраховані за алгоритмом *CRC16*.

У ряді випадків реалізація функцій *ModBus*-Клієнта покладається на операційну систему, а доступ до них в програмі ПЛК відбувається через комунікаційні функції інтерфейсів. Зокрема, це характерно для ПЛК *ОВЕН*. В ряді інших систем – клієнтську сторону на прикладному рівні необхідно повністю прописувати в програмі користувача, а інтерфейс надається лише для обміну з комунікаційним портом. В цьому випадку система надає сервіси відправлення та отримання повідомлення (яке формує та аналізує сама програма користувача) і генерації та перевірки контрольної суми.

3.2.2. Реалізація протоколу ModBus в мережі Ethernet

Комунікаційна архітектура *ModBus TCP/IP* [17] базується на стекові протоколів *TCP/IP* і перш за все призначена для роботи на базі *Ethernet*. Усі *ModBus TCP/IP Клієнти* і *Сервери* підключені до мережі *TCP/IP*; міжмережні пристрої типу мостів, маршрутизаторів або шлюзів для з'єднання мережі *TCP/IP* з послідовними лініями підмереж, що дозволяє обмінюватися даними *ModBus Serial Клієнтськими* і *Серверними* пристроями.Таким чином комунікаційна система *ModBus TCP/IP* дозволяє обмінюватися пристроям не тільки на мережах зі стеком *TCP/IP*, а і з пристроями на послідовних лініях зв'язку (*ModBus RTU/ASCII* або

ModBus+). Як приклад можна навести рис. 3.4, взятий зі специфікації протоколу прикладного рівня [16, 17].

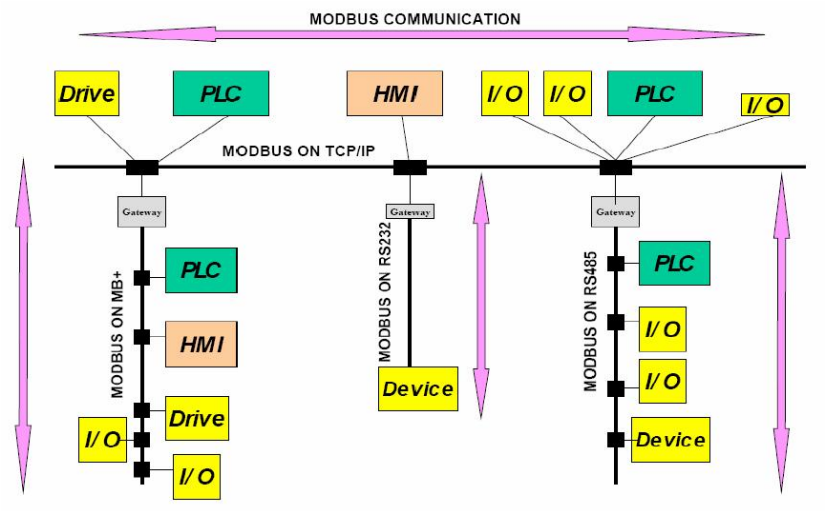


Рисунок 3.4 – Приклад архітектури мережі *ModBus*

Особливості реалізації протоколу *ModBus TCP/IP*. Аналогічно всім мережам *ModBus*, дані мережі використовують *ModBus Application Protocol*. В *ModBus Serial* на канальному рівні до *PDU* додається адреса *Веденого пристрою* та контрольна сума, сам *PDU* не модифікується. Проте в *ModBus TCP/IP* перед попаданням на транспортний рівень до *PDU* (код функції та дані) додається додатковий *MBAP*-заголовок, який складається з полів, які описані в табл. 3.11. Отриманий інкапсулований модуль передається рівню *TCP*.

Відзначимо, що за допомогою поля *UnitID* можна вказати адресу вузла в *ModBus Serial*, наприклад адресу *Веденого* в *ModBus RTU*. Якщо потрібно адресувати вузол, безпосередньо підключений по *TCP/IP*, то UnitID=0. Поле *ProtocolID* використовується для міжсистемного мультиплексування. Так, *TCP* порт для *ModBus*-Серверу має номер 502, який на цей час є стандартним портом протоколу *ModBus TCP/IP*. Наприклад, цей порт використовують контролери *ОВЕН*. Для ідентифікації пристрою,

якому передається запит, вказується його *IP*-адреса. Для ідентифікації *TPDU*, які направляються *ModBus*-Серверу, як прикладному об'єкту, виділений 502-й *TCP*-порт. З деталями функціонування стека *TCP/IP*, а також *Ethernet* можна ознайомитись в літературі по комп'ютерних мережах [18].

Таблиця 3.11 – Структура заголовка модуля *MBAP* в протоколі *ModBus TCP/IP*

Поле	Довжина, байт	Пояснення	Клієнт	Сервер
TransactionID	2	Ідентифікація транзакцій Запитів/Відповідей	Ініціалізує клієнт у Запиті	Копіює із запиту у Відповідь
ProtocolID	2	Тип протоколу 0 = ModBus	Ініціалізує клієнт у Запиті	Копіює із запиту у Відповідь
Length	2	Кількість наступних байтів	Ініціалізує клієнт у Запиті	Ініціалізує сервер у Запиті
UnitID	1	Адреса Веденого вузла	Ініціалізує клієнт у Запиті	Копіює із запиту у Відповідь

3.3. Реалізація транзакцій у протоколі *Modbus* в *ПЛК150*

ModBus – це протокол обміну і розроблений для ПЛК однойменний програмний модуль, який забезпечує роботу ПЛК відповідно до цього протоколу. Для доступу до даних ПЛК із застосуванням протоколу *ModBus* по послідовних інтерфейсах використовують технологію «головний – підпорядкований» («*master – slave*»), при якій лише один пристрій (головний) може ініціювати передачу (зробити запит на читання або запис). Інші пристрої (підпорядковані) передають запитані головним пристроєм дані або проводять запитані дії. Для обміну даними по протоколу *Modbus* в *ПЛК150 ОВЕН* існує два варіанти реалізації: за допомогою бібліотеки *Modbus.lib* або за допомогою налагоджування конфігурації ресурсів ПЛК. В останньому випадку до стандартної конфігурації додаються та налагоджуються програмні модулі *Modbus (Master)* або *Modbus (slave)*, за допомогою яких реалізується обмін з ПЛК в статусі ведучого або веденого пристрою. При

цьому в проекті, створеному в середовищі *CoDeSys* для ПЛК, програма користувача може бути пустою. До складу мережних модулів включені інтерфейси, які самостійно налаштовує користувач. Спочатку розглянемо випадок, де ПЛК виконує функцію *головного* пристрою і взаємодіє з імітатором *підпорядкованого* пристрою – програмою *ModSim32*, потім ПЛК конфігурується як *підпорядкований* пристрій і опитується програмою *CAS Modbus Scanner*. Для розшифровки запитів використовується програма *CAS Modbus RTU Parser*.

3.3.1. Виконання ПЛК150 функції ведучого пристрою в протоколі ModBus

Отже, створіть в середовищі *CoDeSys V2* новий проект, визначіть необхідний таргет-файл та мову реалізації *POU* PLC_PRG. Як цільову платформу оберіть ПЛК *ОВЕН150-I.L*, а мову програмування оберіть *CFC*. Оскільки ПЛК лише опитуватиме ведений пристрій (програму-симулятор *Modbus*-пристрою), вікно з *POU* PLC_PRG залиште пустим. Далі перейдіть у вкладку «*Ресурсы*» і виберіть утиліту «*Конфигурация ПЛК*». У ній через контекстне меню створіть програмний модуль Modbus (Master) [VAR], а в ньому замініть слот з інтерфейсом *RS-232* на слот *RS-485*. Далі налаштуйте параметри інтерфейсу обміну: вкажіть швидкість – 9600, формат кадру – «*8-n-1*» та режим обміну – *RTU*, як це показано на рис. 3.5.

Створіть віртуальну модель підпорядкованого пристрою – Universal Modbus device [VAR] і вкажить його параметри – тут досить вказати адресу пристрою, наприклад «16», решту параметрів можна залишити за умовчанням. На рис. 3.6 наведено параметри модуля підлеглого пристрою.

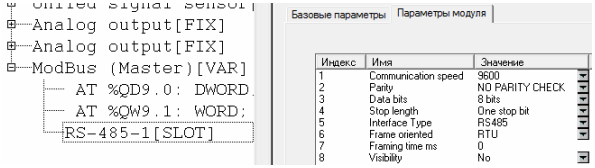


Рисунок 3.5 – Параметри інтерфейсу *RS-485* модуля Modbus (Master) [VAR]

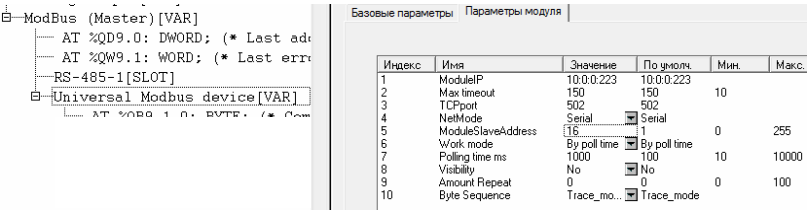


Рисунок 3.6 – Параметри модуля Universal Modbus device [VAR]

Нарешті створіть мережні змінні: вхідний (Register input module [VAR]) та вихідний (Register output module [VAR]) регістри. Вкажіть номери регістрів та функції, які будуть застосовані до них. В результаті отримаєте структуру, яка наведена на рис. 3.7 та 3.8.

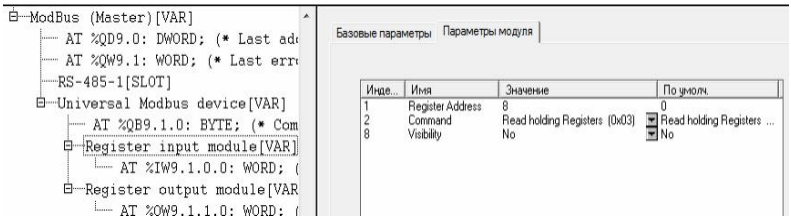


Рисунок 3.7 – Параметри вхідного регістру

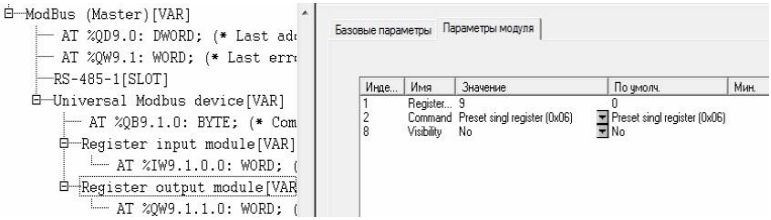


Рисунок 3.8 – Параметри вихідного регістру

Скомпілюйте проект, підключіться до ПЛК за допомогою програмного комунікаційного модуля *CoDeSys Gateway* і завантажте код проекту у ПЛК. Для завантаження використайте інтерфейс *Ethernet*. При необхідності налаштуйте готовий проект. Запустіть ПЛК для виконання програми і не відключайтеся середовищем *CoDeSys* від ПЛК для спостереження за змінними проекту.

Для прийому запитів запустіть на ПК програму моніторингу *COM*-порту *ComRead* та програму розшифрування *Modbus*-запитів *CAS Modbus RTU Parser*. Налаштуйте інтерфейс обміну (швидкість та формат кадру) і далі спостерігайте за прийнятими запитами від ПЛК. На рис. 3.9 показано вікно програми *ComRead* з прийнятим запитом від ПЛК на 7-му *COM*-порті.

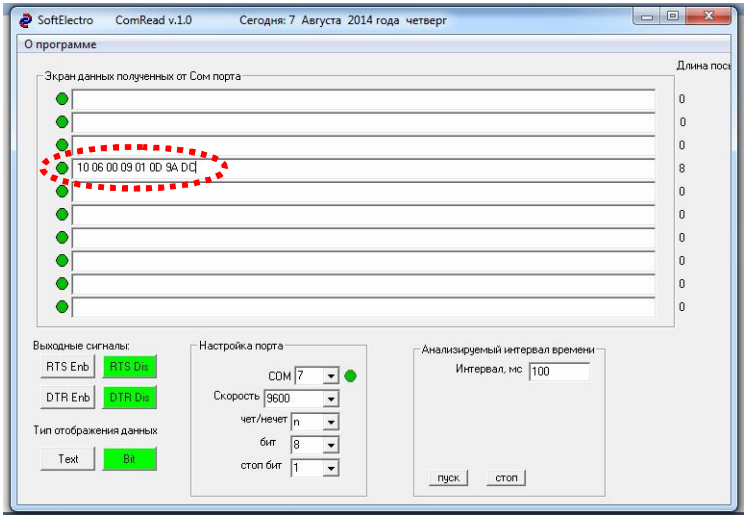


Рисунок 3.9 – Програма моніторингу *COM*-порту

Скопіюйте в буферну пам'ять (Ctrl+C) кодову комбінацію і вставте (Ctrl+V) в поле програми *CAS Modbus RTU Parser* для її розшифрування. Результат розшифрування запиту «10 06 00 09 01 0D 9A DC» показано нижче:

```
Response Analysis
-----------------
Device Address= 16     = 0x10
Function       = 6     = 0x06 = Preset Single Register
Point Address  = 40010
Required value = 269   = 0x01 0D
```

У даному випадку це запит на запис (функція «*06*» – *Preset Single Register*) числа «*269*» у вихідний регістр №9, тобто це друга мережна змінна типу – Register output module [VAR]. Зазначимо, що останні два байти «9A DC» – це контрольна сума *CRC* для режиму передачі *RTU*.

3.3.2. Виконання ПЛК150 функції веденого пристрою в протоколі ModBus

Повторіть дії, які викладені у пункті 3.3.1, але замість модуля ведучого пристрою підключіть модуль веденого пристрою – це ModBus (slave) [VAR]. У властивостях модуля вкажіть його адресу, наприклад «*16*». Далі в цьому модулі, через контекстне меню, створіть комунікаційний інтерфейс RS-485-1 [VAR]. Параметри інтерфейсу встановіть відповідно до пункту 3.3.1, тобто 9600, 8-n-1. Далі додайте в модуль мережні змінні – типу 8 bits [VAR] та Float [VAR]. В результаті буде отримана така конфігурація веденого пристрою, яка показана на рис. 3.10.

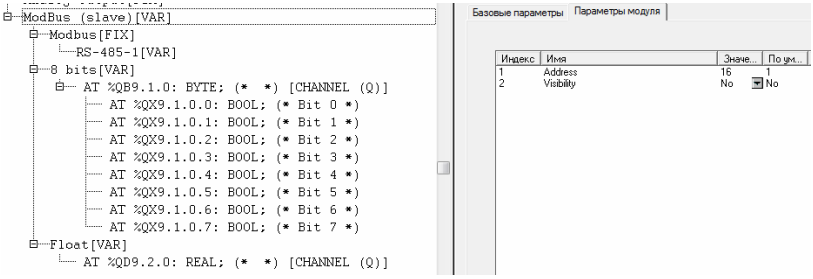


Рисунок 3.10 – Структура веденого пристрою
в конфігурації ресурсів ПЛК

Отже, для читання та запису значень в створеній змінній використовуйте програму *CAS Modbus Scanner*, а для розшифровки запитів – програму *CAS Modbus RTU Parser*.

Відзначимо, що нумерація регістрів в пам'яті веденого ПЛК здійснюється за допомогою так званого правила «вирівнювання», яке описане в документації до ПЛК *ОВЕН* [14, 15]. Також зауважимо, що фізично пам'ять в ПЛК організована за принципом адресування байтів відносно початкового байта (нульового), це так зване «адресування через зсув». На

рис. 3.11 показано розподіл та порядок обчислення адрес в пам'яті ПЛК та відповідність адресам мережних змінних програмного модуля ModBus (slave) [VAR].

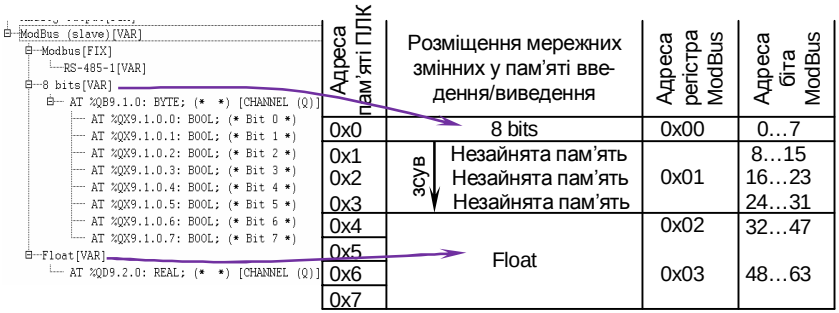


Рисунок 3.11 – Розподіл та порядок обчислення адрес в пам'яті ПЛК

Відповідно до вказаного правила змінна типу *BYTE* (наприклад, регістр зберігання стану обмоток) має адресу 0x0000, а значення параметра типу *REAL* (у даному випадку число з плаваючою комою) зберігається в регістрах з адресами 0x0002 та 0x0003. Тобто, відповідно до правила вирівнювання пропущений регістр з адресою 0x0001. *Це означає таке*: якщо додана наступна змінна має розмір, більш ніж один байт, наприклад, два байти, то вона розміщується в пам'яті ПЛК зі зсувом до парного наступного байта, а якщо чотири байти – то наступна адреса байта повинна ділитися на чотири без залишку. Таким чином, 4-байтова змінна, яка зберігає значення типу *REAL* в пам'яті ПЛК, зсувається на адресу байта 0x0004. Відповідно байти з адресами 0x0001, 0x0002 та 0x0003 залишаються вільними та не використовуються системою виконання ПЛК під час виконання програми. Це не раціонально, але для ПЛК без операційної системи є найбільш простим рішенням.

На рис. 3.12 показаний стан мережних змінних веденого пристрою, які створені у конфігурації ресурсів ПЛК (див. рис. 3.10). В даному випадку перша змінна типу 8 bits [VAR] має значення «*3*», що відповідає знаходженню першого та другого бітів у стані «*true*», а інших бітів – у стані «*false*». Друга змінна типу Float [VAR] має значення «*355,654*».

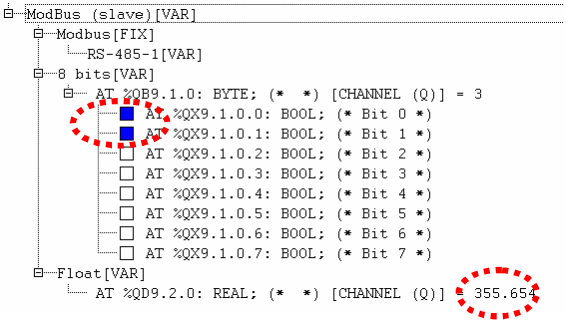


Рисунок 3.12 – Стан мережних змінних в модулі ModBus (slave) [VAR]

Для отримання стану мережних змінних в ModBus (slave) [VAR] запустіть та налаштуйте програму сканування ведених *ModBus*-пристроїв *CAS Modbus Scanner*. На рис. 3.13 показано результат формування запиту на читання регістрів для зберігання мережних змінних, які створені в модулі ModBus (slave) [VAR].

Нижче показаний запит на читання статусу обмоток у вигляді окремих бітів та відповідь на цей запит:

[11:44:41] => Poll: 10 01 00 00 00 08 3E 8D
[11:44:41] <= Response: 10 01 01 03 14 B5 .

Далі показаний запит на виконання установки всіх восьми бітів і квитанція про виконання цієї команди.

[11:47:43] <= Response: 10 0F 00 00 00 08 57 4C
[11:47:43] Write task has completed successfully .

На рис. 3.13 показаний запит на читання регістрів з адресами 0x0002 та 0x0003 і результат читання регістрів, тобто значення числа типу *Float*.

Необхідно дати деякі пояснення щодо адресування регістрів з боку програми *CAS Modbus Scanner*. В програмі застосоване класичне адресування регістрів, яке прийняте в протоколі *ModBus*. Це комбінована адреса в десятковому форматі, яка складається з типу регістра даних та його зсуву відносно першого регістру. Таким чином регістри зберігання значення числа типу *REAL* мають адреси 40003 та 40004. В даному випадку перша

цифра (4) указує, що це регістр з пам'яттю, а 4-розрядний адрес регістру на одиницю більше, ніж адреса в *HEX*-форматі.

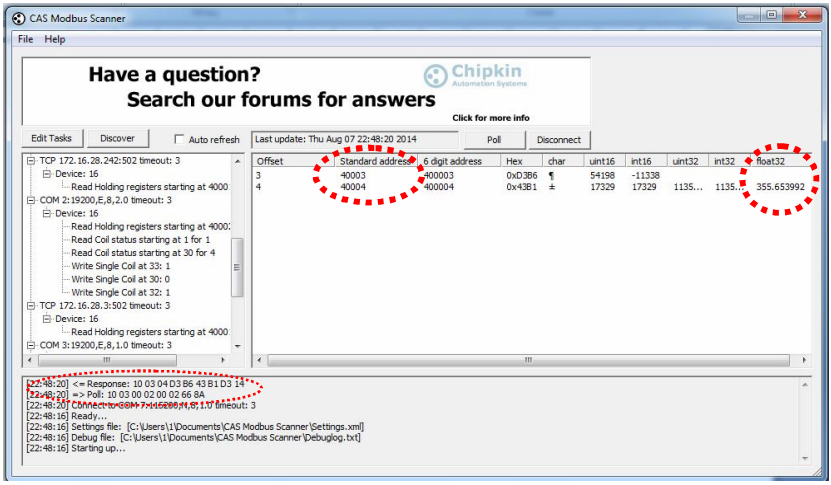


Рисунок 3.13 – Вікно програми сканування ведених *ModBus* пристроїв

Знов скопіюйте в буфер отриману кодову комбінацію і розшифруйте її за допомогою програми *CAS Modbus RTU Parser*:

[12:48:20] => Poll: 10 03 00 02 00 02 66 8A
[12:48:20] <= Response: 10 03 04 D3 B6 43 B1 D3 14 .

Результат розшифрування відповіді на запит буде таким:

Frame Analysis (10 03 04 D3 B6 43 B1 D3 14)
The frame has no errors.
Response Analysis
Device Address =16 =0x10
Function =3 =0x03=Read Holding Registers
Point Count =2 (Implied by byte count=4)
Point Index =1 Value=54198=0xD3_B6. (Address is unknown because the Poll Message was not provided.)
Point Index =2 Value=17329=0x43_B1. (Address is unknown because the Poll Message was not provided.)

Якщо розглядати значення регістрів окремо, то це буде звичайне ціле число без знака (54198 та 17329). Але в нашому випадку було проведено запит на отримання значення числа з плаваючою комою (*REAL*). Відомо, що число типу *REAL* подається в кодованому вигляді за допомогою двійкового коду довжиною 32 розряди. Для подання чисел типу *REAL* використаний міжнародний стандарт *IEEE 754*. Але в протоколі *ModBus* не передбачено передавання чисел, які займають більш ніж один регістр (16 бітів). В таких випадках використовують передавання підряд двох регістрів, які в сумі дають теж 32 розряди. Але в протоколі необхідно визначити порядок відправлення байтів та регістрів у посиланні. Це потрібно прикладним програмам, щоб вони могли правильно інтерпретувати отримані повідомлення. Тому в протоколі *ModBus* значення змінної типу *REAL* певним чином закодоване в двох регістрах. Молодший регістр слідує у відповіді першим, а байти усередині регістра розміщені від старшого до молодшого.

Щоб записати число в стандарті *IEEE 754* або відновити його, необхідно знати три параметри:

- *S* – біт знака числа (31-й біт);
- *E* – зсув експоненти (30–23 біти);
- *M* – залишок від мантиси (22–0 біти).

Це цілі числа, записані у форматі *IEEE 754* в двійковому вигляді. Наведемо формулу для отримання десяткового числа з числа у форматі *IEEE754* одинарної точності:

$$F=(-1)^{S}\cdot 2^{(E-127)}\cdot(1+M/2^{23}),$$

де F – десяткове число. У табл. 3.12 показані результати перетворення значень регістрів у число з плаваючою комою (див. рис. 3.13).

Таблиця 3.12 – Приклад розшифрування повідомлення

У таблиці наведено десяткове число 355,654 у форматі *IEEE754*			
1 біт	8 бітів	23 біти	IEEE 754
0	100_0011_1	011_0001_1101_0011_1011_0110	43B1_D3B6 (HEX)
0 (dec)	135 (dec)	1810432 (dec)	355,654
Знак числа	Зміщена експонента	Залишок від мантиси	Число у форматі *DEC*

3.4. Основні принципи формування запитів і відповідей у протоколі *OWEN*

Протокол *OWEN* [19] був розроблений для обміну інформацією пристроїв ОВЕН між собою і з ПК в мережі *RS-485*. Протокол *OWEN* має зручну організацію для конфігурування приладів. Тому усі модулі введення-виведення та локальні регулятори серії ТРМ конфігуруються за допомогою програм-конфігураторів, які використовують протокол *OWEN*.

Отже, *OWEN* – це протокол обміну і розроблений для ПЛК однойменний програмний модуль, який забезпечує роботу ПЛК відповідно до цього протоколу. Для доступу до даних ПЛК із застосуванням протоколу *OWEN* по послідовних інтерфейсах використовують технологію «головний - підпорядкований» («*master - slave*»), при якій лише один пристрій (головний) може ініціювати передачу (зробити запит). Інші пристрої (підпорядковані) передають запитані головним пристроєм дані або проводять запитані дії.

Для обміну даними по протоколу *OWEN* в *ПЛК150* існує два варіанти реалізації: за допомогою бібліотеки *OwenNet.lib* або за допомогою конфігурування ресурсів ПЛК. В останньому випадку до стандартної конфігурації додаються програмні модулі *Owen (Master)* або *OWEN (Slave/Spy)*, за допомогою яких реалізується обмін даними з ПЛК в статусі ведучого або веденого пристроїв. При цьому в проекті, який створений в середовищі *CoDeSys* для ПЛК, програма користувача може бути пустою. До складу вказаних модулів включені інтерфейси, які налаштовує користувач. Далі користувач додає потрібні змінні для обміну даними.

3.4.1. Виконання ПЛК150 функції ведучого пристрою в протоколі OWEN

Отже, створіть в середовищі *CoDeSys* проект, визначіть необхідний таргет-файл та мову реалізації головного *POU* PLC_PRG. Як цільову платформу виберіть *ОВЕН150-I.L*, а мову програмування виберіть *CFC*. Оскільки ПЛК лише опитуватиме модуль введення/виведення, вікно *POU* PLC_PRG залиште пустим. Далі перейдіть у вкладку «Ресурсы» і виберіть утиліту «Конфигурация ПЛК». Далі, через контекстне меню, створіть програмний модуль Owen (Master) [VAR], а в ньому замініть слот з

інтерфейсом *RS-232* на *RS-485* з такими параметрами інтерфейсу: швидкість – 9600, формат кадру – 8-n-1 і режим передачі – *ASCII*, як це показано на рис. 3.14.

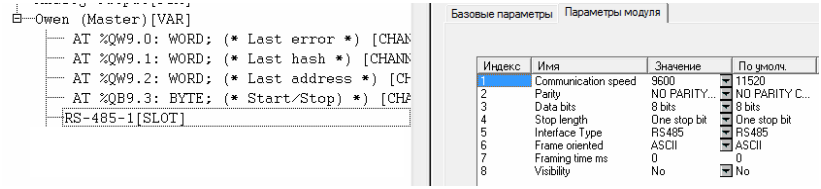


Рисунок 3.14 – Параметри інтерфейсу *RS-485* модуля Owen (Master) [VAR]

Далі, через контекстне меню, додайте мережні змінні. Наприклад, змінну для читання з пристрою поточного значення параметра (PV) і змінну для запису у пристрій значення уставки (SP). Встановіть параметри мережних змінних, вкажіть адресу веденого пристрою, наприклад 16, *HASH*-имя змінних – PV та SP, а також тип змінних – Float. Решту параметрів залиште за умовчанням. На рис. 3.15 та 3.16 показано параметри мережних змінних ведучого пристрою *Owen (Master)*.

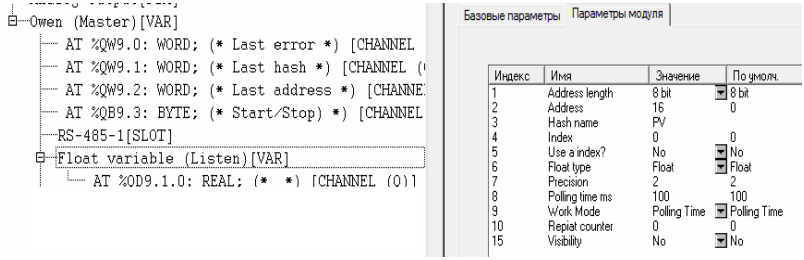


Рисунок 3.15 – Параметри мережної змінної *PV*

Для зручності введення уставок та відображення значення змінних створіть візуалізацію проекту. Скомпілюйте проект, підключіться до ПЛК і завантажте у ПЛК код проекту. Для завантаження використовуйте інтерфейс *Ethernet*. Запустіть ПЛК на виконання програми користувача і не відключайтеся середовищем *CoDeSys* від ПЛК для спостереження за змінними проекту і введення значення уставки.

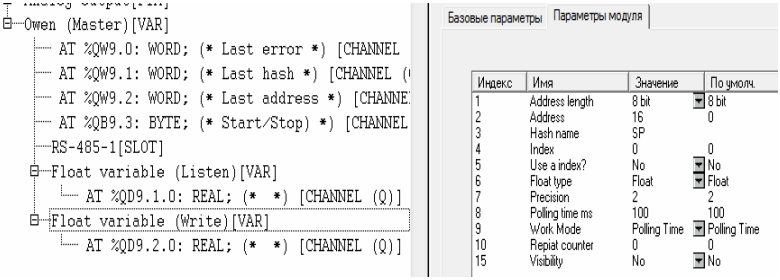


Рисунок 3.16 – Параметри мережної змінної *SP*

Для приймання запитів запустіть на ПК програму моніторингу *COM*-порту *ComRead*. Виберіть потрібний *COM*-порт і налаштуйте його параметри відповідно до налаштувань комунікаційного модуля *RS-485* в проекті для ПЛК, а саме: швидкість – *9600*, формат кадру – *8-n-1* і режим передачі – *ASCII*. На рис. 3.17 показано прийняті кадри запитань з боку контролера.

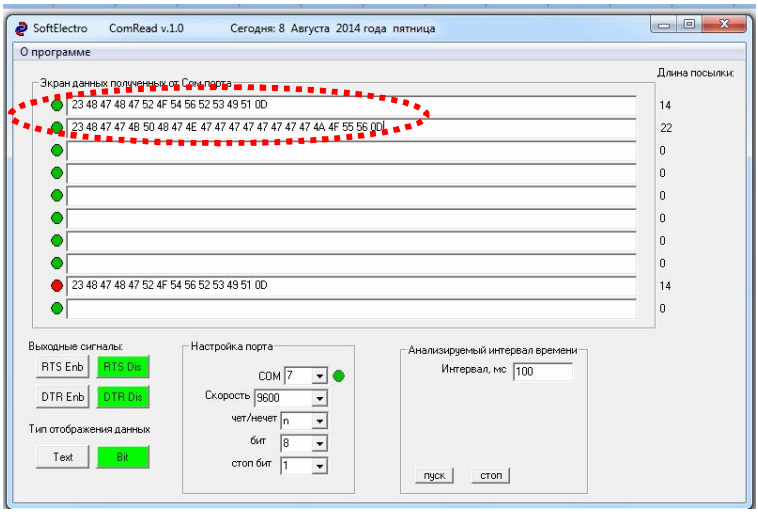


Рисунок 3.17 – Параметри налаштування мережної змінної *SP*

Як видно з рис. 3.17, головний пристрій в ПЛК формує два запити, в точній відповідності з його структурою, описаною раніше (рис. 3.15 та

3.16). Оскільки програма *ComRead* не проводить аналіз кадрів, скористаємося для їх розшифрування офіційним описом протоколу *OWEN*, який можна скачати з сайту [19].

Отже, отримано два кадри:

№ 1 – 23 48 47 48 47 52 4F 54 56 52 53 49 51 0D – усього 14 байтів

№ 2 – 23 48 47 47 4B 50 48 47 4E 47 47 47 47 47 47 47 47 4A 4F 55 56 0D – усього 22 байта.

Обидва кадри містять два службові символи-роздільники кадрів. Це символ початку кадру – 23 (#) та символ кінця кадру – 0D (<*CR*> – повернення каретки). Відкинемо їх, оскільки вони не входять в контрольну суму *CRC*. Далі, відповідно до таблиці в описі протоколу *OWEN* перетворимо отримані символи по схемі «*ASCII-символ – у тетраду*», потім «склеїмо» тетради в байти і перетворимо байти в прикладний рівень, тобто отримаємо адресу, *HASH*-ім'я та іншу інформацію.

Результат розшифрування кадра №1 на рис. 3.17 представлений в табл. 3.13.

Таблиця 3.13 – Порядок розшифрування кадра №1

<table>
<tr><td>48</td><td>47</td><td>48</td><td>47</td><td>52</td><td>4F</td><td>54</td><td>56</td><td>52</td><td>53</td><td>49</td><td>51</td></tr>
<tr><td>0001</td><td>0000</td><td>0001</td><td>0000</td><td>1011</td><td>1000</td><td>1101</td><td>1111</td><td>1011</td><td>1100</td><td>0010</td><td>1010</td></tr>
<tr><td colspan="2">10</td><td colspan="2">00010000</td><td>B</td><td>8</td><td>D</td><td>F</td><td>B</td><td>C</td><td>2</td><td>A</td></tr>
<tr><td colspan="2" rowspan="2">16_{hex}</td><td colspan="2" rowspan="2">біт запиту
встановлений
кільк. байт поля даних (n–2=0)</td><td colspan="3">B8</td><td colspan="3">DF</td><td colspan="3">BC</td><td colspan="2">2A</td></tr>
<tr><td colspan="4">B8DF – HASH-код змінної з ім'ям PV = <50><62></td><td colspan="4">CRC – контрольна сума</td></tr>
</table>

Результат розшифрування кадра №2 на рис. 3.17 представлений в табл. 3.14.

Таблиця 3.14 – Порядок розшифрування кадра №2

<table>
<tr><td>48</td><td>47</td><td>47</td><td>4B</td><td>50</td><td>48</td><td>47</td><td>4E</td><td>47</td><td>47</td><td>47</td><td>47</td></tr>
<tr><td>0001</td><td>0000</td><td>0000</td><td>0100</td><td>1001</td><td>0001</td><td>0000</td><td>0111</td><td>0000</td><td>0000</td><td>0000</td><td>0000</td></tr>
<tr><td colspan="2">10</td><td colspan="2">000000100</td><td>9</td><td>1</td><td>0</td><td>7</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="2" rowspan="2">16_{hex}</td><td colspan="2" rowspan="2">біт запиту
зброшений
кільк. байт поля даних (6–2=4)</td><td colspan="4">91</td><td colspan="4">07</td><td colspan="4">00</td><td colspan="2">00</td></tr>
<tr><td colspan="4">9107 – HASH-код змінної з ім'ям SP = <56><50></td><td colspan="6">Поле даних – значення змінної у форматі IEEE 754</td></tr>
</table>

Продовження таблиці 3.14

47	47	47	47	4A	4F	55	56
0000	0000	0000	0000	0011	1000	1110	1111
00		00		3	8	E	F
Поле даних – значення змінної у форматі *IEEE 754*				38		EF	
				CRC – контрольна сума			

Приклад запису ненульового значення уставки (45,25) до пристрою показаний нижче, де кутовими дужками визначено закодоване значення уставки:

23 48 47 47 4A 50 48 47 4E < 4B 49 4A 4C 47 47 > 54 51 4E 4E 0D .

Відмітимо, що в даному випадку змінна має тип *PIC-Float*, тобто складається з трьох байтів з укороченою мантисою, що незначно впливає на 5-й знак після коми. Така точність при передачі даних цілком прийнятна.

3.4.2 Виконання ПЛК150 функції веденого пристрою протоколу OWEN

Повторіть дії, які викладені в пункті 3.*4.1*, але замість головного пристрою підключіть підлеглий – OWEN (slave) [VAR]. У властивостях модуля вкажіть його адресу – 16 та ім'я веденого пристрою, наприклад, plc150, як це показано на рис. 3.18.

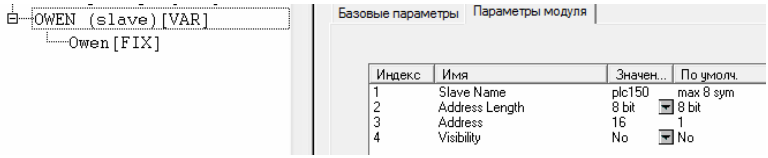


Рисунок 3.18 – Параметри модуля OWEN (Slave) [VAR]

Далі, через контекстне меню, створіть в модулі комунікаційний інтерфейс RS-485-1 [VAR]. Параметри інтерфейсу встановіть такими: швидкість – *9600*, формат кадру – *8-n-1*, режим обміну – *ASCII*. На рис. 3.19 показані параметри інтерфейсу *RS-485* веденого пристрою.

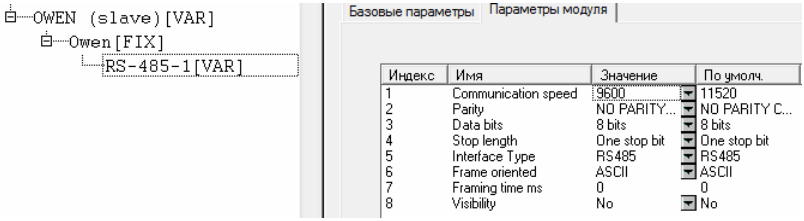


Рисунок 3.19 – Параметри інтерфейсу *RS-485* модуля OWEN (Slave) [VAR]

Далі додайте в модуль дві мережні змінні – типу Float variable (Listen) [VAR] і типу Unsigned variable (Listen) [VAR] з субмодулем 2 byte [VAR]. В даному випадку ПЛК імітуватиме роботу веденого пристрою, наприклад модуля введення аналогових сигналів *MBA*, і передавати за запитом значення і тип датчика на першому вимірювальному каналі. Для цих цілей в параметрах мережних змінних потрібно вказати відповідні *HASH*-імена цих змінних – це rEAd та in-t, а також адресу першого каналу – 16 (співпадає з базовою адресою пристрою), а інші параметри залиште за умовчанням. В результаті буде отримана така конфігурація веденого пристрою, яка показана на рис. 3.20 та 3.21.

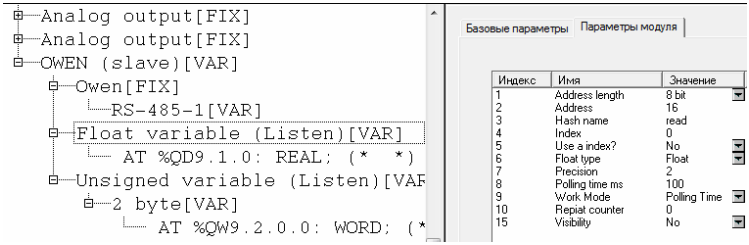


Рисунок 3.20 – Параметри налаштування мережної змінної *read*

Для читання значень у веденому пристрої запустіть програму *OWEN Test Console*, яка виконує функцію ведучого пристрою. Після запуску програми з’явиться її консоль, де необхідно провести налаштування послідовного порту, вибрати тип протоколу, вказати ім’я змінної.

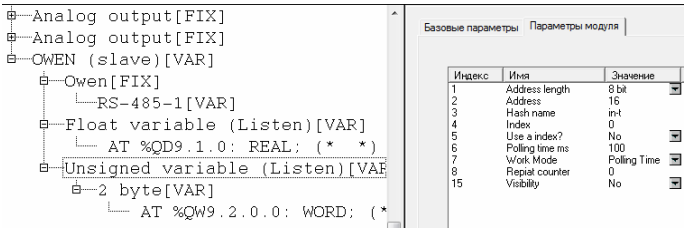


Рисунок 3.21 – Параметри налаштування мережної змінної *in-t*

Отже, в зоні «*Прибор*» натисніть на кнопку «*Настройка связи*» і в діалоговому вікні, що відкрилося, «*Установка параметров обмена*» вкажіть тип перетворювача інтерфейсів – автоматичний, виберіть потрібний *COM*-порт і зробіть необхідні налаштування в зоні «*Формат данных*»: *9600, 8-n-1*. Також у зоні «*Сетевые параметры*» введіть адресу веденого пристрою – 16. Інші параметри залиште за умовчанням. Для запам'ятовування налаштувань натисніть на кнопку «*ОК*». Далі оберіть тип пристрою – «*ОВЕН стандарт*», оскільки в *ПЛК150* використовується програмний модуль OWEN (Slave) [VAR] як імітатор веденого пристрою. У зоні «*Параметр*» в полі «*Тип*» – RAW, в полі номера канала – 0 і в полі «*Имя*» введіть ім'я змінної – rEAd.

Якщо натиснути на кнопку «*Получить*», то можна отримати значення на першому вимірювальному каналі. В даному випадку – це код [43 E2 51 3E]. На рис. 3.22 показані відповідні налаштування програми. Якщо необхідно отримати розшифроване значення, то в зоні «*Параметр*» замініть тип *RAW* на тип *FLOAT IEEE (PIC)*, який відповідає стандарту *IEEE 754* з відкинутим молодшим байтом мантиси (див. табл. 3.15). Дійсне значення параметра буде складати 452.6347 (перевірте самостійно). В програмі за допомогою кнопки «*Записать*» можна записати значення параметра, наприклад уставку (SP) у ведений пристрій.

Для прикладу запишіть нове значення до ПЛК, а потім прочитайте його у «сирому» (код значення) і розшифрованому вигляді (число). Зверніть увагу, що число приходить з більшою кількістю знаків після коми. Це особливість формату стандарту *IEEE 754*.

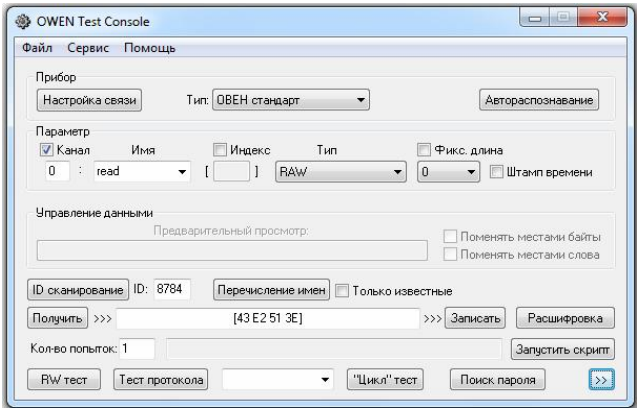


Рисунок 3.22 – Параметри налаштування програми для зчитування значення змінної *read* на першому каналі пристрою

Таблиця 3.15 – Приклад розшифрування коду параметра

У таблиці наведено десяткове число 452.6347 у форматі *IEEE754*			
1 біт	8 бітів	23 біти	IEEE 754
0	100_0011_1	110_0010_0101_0001_0011_1110	43E2_513E (HEX)
0 (dec)	135 (dec)	6 443 326 (dec)	452.6347
Знак числа	Зміщена експонента	Залишок від мантиси	Число у форматі *DEC*

Існує ще один варіант подання числа типу *FLOAT*. Це знакове число з односторонньою десятковою крапкою в протоколі *OWEN*, яке записується формулою:

$$N = (-1)^{S} \bullet 10^{(-\,Exponent)} \bullet Mantissa,$$

де S – визначає знак і передається одним бітом (старшим);

Exponent – десятковий порядок числа, який передається числом без знака, визначає положення десяткової коми (0 – для цілого числа, 1 – для числа з десятковими частками тощо), передається трьома наступними бітами після біта знака;

Mantissa – ненормалізована мантиса, біти, які залишились (довжина мантиси визначається розміром поля даних). Сумарна довжина поля даних повинна складати ціле число байтів.

Мантиса може виражатися як у двійковому, так і у двійково-десятковому вигляді (код BCD, див. стор. 127). Наприклад, число «-10,38» може бути закодоване так:

- двійково-десяткове подання мантиси – 0xA01038;

A	0	1	0	3	8
1 010	0000	0001	0000	0011	1000
S=1 EXP=2	Доповнює поле до цілого числа байтів	1	0	3	8

- двійкове подання мантиси (на один байт коротше) – 0xA40E.

A	4	0	E
1 010	0100	0000	1110
S=1 EXP=2	1038		

3.5. Основні принципи формування запитів і відповідей в протоколі *DCON*

Протокол *DCON* хоча не належить до стандартних, але достатньо популярний завдяки застосуванню видалених модулів введення/виведення, які розроблені компаніями *ICP DAS* (http://www.icpdas.com) і *Advantech* (http://www.advantech.com). Він реалізує символьний обмін даними в послідовних інтерфейсах. Крім того, протокол *DCON* підтриманий в деяких приладах ОВЕН, наприклад, в ПЛК 100-ої серії та модулях введення/виведення: МВА, МДВВ, МВУ, Мх110, а також в інших приладах ОВЕН. Популярність протоколу *DCON* обумовлена відсутністю необхідності в спеціалізованих мікросхемах для реалізації стека протоколів, що істотно знижує собівартість пристроїв, а, отже, ціну для кінцевого споживача. З іншого боку, у системного інтегратора зменшуються витрати на навчання, оскільки застосування протоколу достатньо просте.

Для аналізу кадрів протоколу *DCON* від веденого пристрою можна використовувати програму OWEN Test Console, яка емулює роботу го-

ловного пристрою *DCON*. Для цього досить налаштувати роботу модуля на роботу за протоколом *DCON* та опитувати модуль вказаною програмою. Конфігурація модулів *MBA* і *МДВВ* для роботи за протокоом *DCON* проводиться за допомогою програм «*Конфигуратор MBA8*» і «*Конфигуратор МДВВ*», які взаємодіють з модулями за допомогою ПІ. Контролери 100-ї серії ОВЕН також можуть виконувати функцію головного пристрою за допомогою відповідного програмного модулю протоколу *DCON* – DCON (Master) [VAR].

На рис. 3.23 та 3.24 показані приклади з'єднання ПЛК та ПК з приладами ОВЕН (*МДВВ* та *MBA*) для реалізації протоколу *DCON*.

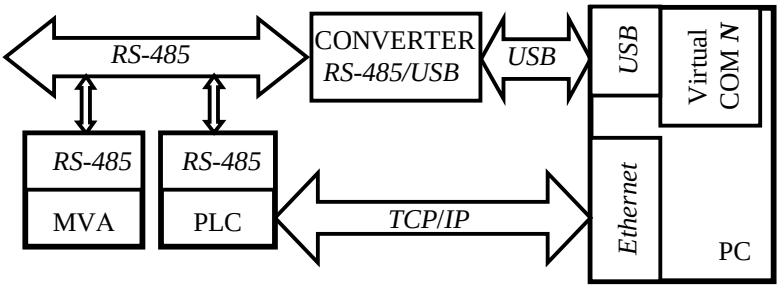


Рисунок 3.23 – Схема з'єднань ПК, *ПЛК150* та модуля *MBA* для реалізації обміну за протоколом *DCON*

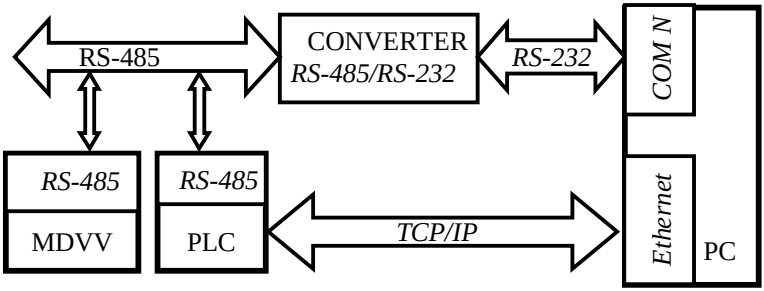


Рисунок 3.24 – Схема з'єднань ПК, *ПЛК150* та модуля *МДВВ* для реалізації обміну за протоколом *DCON*

Даний протокол використовує лише фізичний та прикладний рівні моделі *OSI*. На фізичному рівні використовується пряме двійкове кодування *ASCII*-символів. Вимоги до середовища передачі визначаються стандартом на інтерфейси *RS-485* і *RS-232*. Також можлива робота за протоколом *DCON* через модем.

У протоколі *DCON* застосовується метод доступу до каналу зв'язку «ведучий – ведений». У мережі може бути лише один ведучий і до 255 ведених пристроїв. У протоколі *DCON* при організації опитування пристроїв створюється рядок запиту, і, при її посиланні, опитуваний пристрій може повернути два варіанти відповіді: *правильну відповідь*, якщо команда розпізнана і є дані; і *неправильну відповідь*, якщо команда не розпізнана і даних немає.

Для підвищення надійності передачі даних на фізичному рівні використовується метод обчислення контрольної суми. Оскільки канальний рівень в протоколі не передбачений, то помилки передачі можуть бути виявлені тільки на прикладному рівні, тобто безпосередньо в програмі користувача.

Кадр прикладного рівня протоколу *DCON* має такий вигляд:

Роздільник	Адреса	Команда	Блок даних	Контрольна сума
1 байт	1 байт	1…5 байтів	1…256 байтів	1 байт

Вся інформація, що міститься в кадрі, включаючи адресу модуля і дані, передається в *ASCII-кодах* символів. Кожен кадр починається з роздільника. Як роздільник можуть бути використані символи: $, #, %, @, *; у відповідях веденого пристрою використовуються символи ~, !, ?, >. За деякими командами можуть слідувати дані, але їх може і не бути. Контрольна сума, яка складається з двох символів, може бути відсутньою. Кожен кадр повинен закінчуватися символом повернення каретки <CR> (*ASCII*-код $0D_{hex}$). Рядки формату відповіді можуть не задаватися, якщо пристрій не відповідає на запити. Рядок формату відповіді містить певні **символи** і **спецкоманди**.

Символ – будь-який символ, окрім службових, до яких відносяться символи $, [та]. При необхідності вивести службовий символ як звичайний, він вводиться в рядок двічі підряд.

Спецкоманда має такий формат: **[{модифікатор} дія].**

Модифікатор – кількість символів, що обробляються дією. Є десятковим цілим числом. Може бути у всіх дій, окрім обчислення контрольних сум. Наявність модифікатора необов'язкова, значення за умовчанням дорівнює одиниці.

Дія – відображається в рядку спецкоманди одним з таких символів – D, H, F, S, *, +, %. Регістр символів значення не має. Символи відповідають таким видам дій:

а) D – подає змінну, яка передається в *ASCII*-коді в десятковому форматі (без знака), або перетворює *ASCII*-рядок з десяткового формату (без знака) в змінну, що приймається. Кількість символів задається модифікатором;

б) H – подає змінну, яка передається в *ASCII*-символах у *HEX*-форматі, або перетворює *ASCII*-рядок з *HEX*-формату у змінну, що приймається. Кількість символів задається модифікатором;

в) F – подає змінну, яка передається в *ASCII*-символах у десятковому форматі зі знаком, роздільником цілої та дробової частини числа (крапкою). Рядок має фіксоване число символів, яке задане модифікатором. Для змінних, що приймаються, проводить зворотне перетворення з *ASCII*-рядка у число;

г) S – здійснює пряме копіювання зі змінної типу «рядок», яка передається, у рядок запиту числа символів, яке задане модифікатором або зворотне копіювання з рядка відповіді у змінну рядкового типу, що приймається;

д) * – задає в рядку відповіді набір символів, які треба пропустити. Кількість символів задається модифікатором;

е) + – вставляє в рядок запиту контрольну суму або отримує її в рядку відповіді. Контрольна сума обчислюється шляхом складання з переповнюванням по модулю 256. Дана дія не має модифікатора;

ж) % – вставляє в рядок запиту контрольну суму або отримує її в рядку відповіді. Контрольна сума обчислюється за 8-бітовим поліномом (*CRC*). Дана дія не має модифікатора.

Контрольна сума (*CHK*) складається з двох символів *HEX*-формату (у суму не входить код символу <CR>) і дозволяє виявити помилки в командах, посланих ведучим пристроєм, а також у відповідях веденогопристрою. Контрольна сума передається безпосередньо перед символом <CR> і має дорівнювати сумі кодових значень усіх *ASCII*-символів команди. Якщо сума більша значення FF_{hex}, то як контрольна сума використовується лише її молодший байт.

Наприклад, якщо потрібно переслати веденому пристрою команду $052<CR>, то контрольна сума *ASCII*-кодів символів команди (символ <CR> не рахується) рівна:

$$\text{“\$”+”0”+”5”+”2”}=24_{hex}+30_{hex}+35_{hex}+32_{hex}=BB_{hex}.$$

Таким чином, перед символом <CR> у команду треба додати код BB_{hex}, тоді команда $052 буде такою: $052BB<CR>.

Якщо відповідь модуля на цю команду без контрольної суми буде, наприклад, !05200600<CR>, то сума *ASCII*-кодів символів цієї команди рівна

$$\text{“!”+”0”+”5”+”2”+”0”+”0”+”6”+”0”+”0”}=$$
$$=21_{hex}+30_{hex}+35_{hex}+32_{hex}+30_{hex}+30_{hex}+36_{hex}+30_{hex}+30_{hex}=1AE_{hex}.$$

У цьому випадку відповідь модуля з контрольною сумою буде такою: !05200600C0AE<CR>, де передостанній байт «C0» означає, що встановлений режим обміну з контрольною сумою.

При написанні програми прикладного рівня використовується набір стандартних команд, приклади яких наведені в табл. 3.16.

Команди протоколу *DCON* діляться на чотири типи:

- команди модулів аналогового введення;
- команди модулів аналогового виведення;
- команди дискретного введення/виведення;
- команди лічильників та таймерів.

Таблиця 3.16 – Приклади команд протоколу *DCON*

Команда	Відповідь	Опис
%AANNTTCCFF	!AA	Встановлює адресу, діапазон вхідної напруги, швидкість обміну, формат даних, контрольну суму
#AA	>(Data)	Повертає всі вхідні значення аналогових каналів для заданого модуля
#AAN	>(Data)	Повертає вхідне значення аналогового каналу номер (*N*) для заданого модуля
$AA0	!AA	Виконує калібрування аналогового модуля для компенсації помилки коефіцієнта передачі
$AA1	!AA	Виконує калібрування аналогового модуля для компенсації помилки зсуву нуля
$AA2	!AATTCCFF	Повертає параметри конфігурації модуля з вказаною адресою

Розглянемо приклади формування запитів та аналізу відповідей за протоколом *DCON*.

1) Розглянемо приклад використання команди #AA для отримання даних з модуля аналогового введення (наприклад, 8-канального).

Синтаксис команди запиту буде таким:

AA [CHK] <CR>,

де AA – адреса модуля (від 00_{hex} до FF_{hex}), [CHK] – контрольна сума, <CR> – повернення каретки.

Відповідь модуля на команду матиме такий формат:

> (Data) [CHK] <CR>,

якщо команда прийнята, правильно розшифрована і виконана. Тут **>** – символ-роздільник при команді, яка виконана, (Data) – дані, що вимірю-

ються, [CHK] – контрольна сума, <CR> – повернення каретки. Якщо мали місце синтаксичні помилки або помилка зв'язку, то відповіді від модуля не буде.

Наприклад, для опитування модуля з адресою «26_{10}» дана команда буде такою: # 1A [CHK] <CR>.

Відповідь модуля на команду може бути такою:

>+1.2345+0.3456+0.0001+2.5000+1.2345+0.3456+

0.0001+2.5000 [CHK] <CR>.

У відповіді наведено вісім значень напруги на восьми входах модуля введення аналогових сигналів.

2) Приклад установки вихідного значення модуля аналогового виведення *IPC-7021* з періодичністю в одну секунду і при необхідності зміни значення.

Синтаксис команди запиту буде таким:

AA (дані) [CHK] [CR],

де # – роздільник; AA – адреса пристрою; (дані) – вихідне значення (5 чисел + знак + кома), усього – сім символів, [CHK] – контрольна сума, [CR] – повернення каретки.

Якщо команда прийнята і виконана, то відповідь модуля на команду матиме такий формат:

! [CHK] [CR],

де ! – роздільник у випадку успішного виконання команди; [CHK] – контрольна сума, [CR] – повернення каретки.

3.5.1. Виконання ПЛК150 функції ведучого пристрою в протоколі DCON

Нижче наведений приклад конфігурування в середовищі *CoDeSys* ресурсів ПЛК модуля DCON (Master) [VAR] з підключеним до нього модулем Universal DCON device [VAR]. Тут проводиться запис значення на вихід модуля *IPC-7021* з адресою 24_{10} (18_{hex}). Дані, що посилаються в модуль *IPC 7021*, задаються у вихідний змінної *output_1* типу *Float*

(*REAL*). Вікно конфігурації програмного модуля в *ПЛК150*, який налаштований для періодичного запису вихідних значень у модуль *IPC-7021*, показане на рис. 3.25.

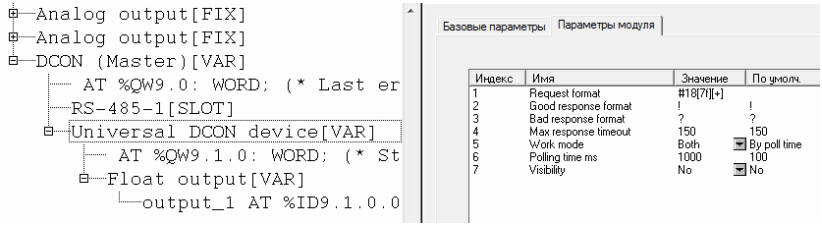


Рисунок 3.25 – Схема налаштувань *ПЛК150* та модуля *МДВВ*

Формат рядка запиту буде таким:

18 [7f] [+],

де # – символ роздільника команди; 18_{hex} – адреса пристрою у *HEX*-форматі (***для букв використовується верхній регістр!***); [7f] – спецкоманда, яка вказує на те, що сім символів запиту мають бути сформовані у вигляді числа у форматі « [знак] число число число число . число », причому дані мають бути взяті з вихідної змінної, яка повинна мати формат *float*; [+] – спецкоманда підрахунку і додавання в кінець запиту контрольної суми «за модулем 256».

Увага! Символ повернення каретки в посилку вставляється автоматично!

Формат позитивної відповіді буде – !. Формат негативної відповіді – ?. У цих випадках дані символи не містять ніякої іншої інформації.

3.5.2. Налаштування зв'язку з модулями введення/виведення ОВЕН за протоколом DCON

Модулі введення/виведення ОВЕН, в яких підтриманий протокол *DCON*, перед підключенням до РСУ мають бути налаштовані на відповідний протокол роботи. Ці модулі працюють в мережі *RS-485* за протоколом *DCON* лише за наявності в ній головного пристрою (ПК або ПЛК). Раніше зазначалося, що конфігурування модулів здійснюється за прото-

колом *OWEN* за допомогою програм-кофігураторів. Для інтеграції модулів до складу РСУ необхідно:

- по-перше, за допомогою програми-конфігуратора налаштувати інтерфейс та протокол обміну даними з ПК або ПЛК;
- по-друге, провести конфігурування входів та виходів відповідно до типу датчиків, що підключаються, і виконавчих механізмів.

Зауважимо, оскільки конфігурування модуля здійснюється лише за протоколом *ОВЕН*, то для переходу пристрою на роботу за протоколом, який вказаний у параметрі «Prot», після завантаження конфігурації необхідно вимкнути та ввімкнути модуль.

Отже, для підготовки пристрою до роботи в РСУ необхідно через ПІ підключити його до ПК та подати живлення. Конфігурування включає налаштування мережних параметрів і мережного інтерфейсу (конфігурація пристрою – це повний набір значень параметрів пристрою, який визначає його роботу). Після запуску програми-конфігуратора у вікні налаштування мережних параметрів програми необхідно задати параметри для зв'язку з пристроєм по інтерфейсу *RS-485*, а саме: швидкість – *9600*, формат кадру – *8-n-1*, адреса пристрою – 16 і номер *COM*-порту. Далі для зв'язку пристрою з ПЛК або ПК у складі РСУ необхідно вибрати необхідний протокол обміну (*DCON*), а також налаштувати вимірювальні входи. Після закінчення конфігурування модуля збережіть та завантажте проект з конфігурацією у пристрій, заздалегідь встановивши з ним зв'язок. Перезавантажте пристрій для переходу на роботу за протоколом *DCON*.

Для опитування модулів ОВЕН за протоколом *DCON* з боку ПК застосовується програма *OWEN Test Console* та використовується ПІ. Запустіть програму *OWEN Test Console*, яка виконує функцію ведучого пристрою мережі. Для читання значень у веденому пристрої виберіть протокол, потім тип ПІ, налаштуйте параметри інтерфейсу і вкажіть адресу модуля. На рис. 3.26 показані параметри обміну для опитування модулів за протоколом *DCON*.

Після налаштування параметрів обміну виберіть тип роздільника – перший байт (виберіть символ-роздільник «#») та опитуваний канал (наприклад, 4-й канал) і натисніть на кнопку «*Получить*». Відзначимо, що

нумерація каналів у модулі починається з нульового. У полі відповіді спостерігайте за повідомленнями модуля.

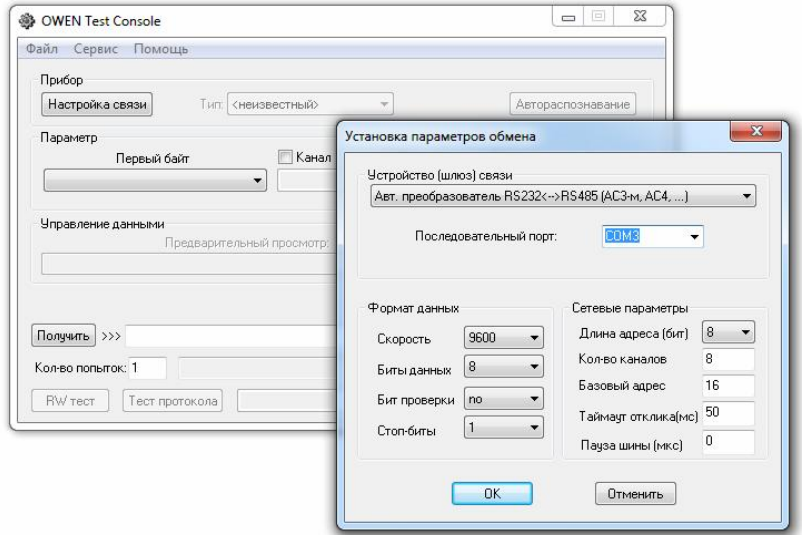


Рисунок 3.26 – Налаштування інтерфейсу програми *OWEN Test Console*

Функціонування *ПЛК150* за протоколом *DCON* в середовищі *CoDeSys* реалізовано лише в режимі ведучого пристрою і реалізується при додаванні в конфігурацію ресурсів ПЛК програмного модуля *DCON* (*Master*), який по вибраному комунікаційному інтерфейсу обмінюється даними з іншими пристроями. При цьому в проекті програма користувача може бути пустою.

До складу модуля включені лише послідовні інтерфейси, які користувач налаштовує самостійно. Крім того, користувач додає потрібні змінні для обміну даними. В процесі роботи ПЛК відправляє запити веденим пристроям по інтерфейсу *RS-485* на читання параметрів каналів й установку значень виходів. Зауважимо, що в процесі роботи *ПЛК150* за протоколом *DCON* є три варіанти роботи модуля: без розрахунку контрольних сум, з розрахунком контрольних сум шляхом складання значень всіх символів (за модулем «256») і з розрахунком контрольних сум 8-бітових

символів з *CRC*. Варіант роботи користувач обирає відповідно до того, який варіант розрахунку контрольної суми використовується у пристрої, що запитується.

Використовуються такі алгоритми перетворення даних у модулі:

- ***при формуванні запиту*** **–** всі символи поза спецкомандами копіюються в рядок запиту без зміни, спецкоманди замінюються на значення змінних (*вихідних*), які передаються. Значення змінних кодуються у форматі, який заданий дією, число символів відповідає модифікатору;
- ***при розборі відповіді*** **–** всі символи поза спецкомандами порівнюються з відповідними позиціями відповіді і при знаходженні відмінності виробляється повідомлення про помилку. Дані в позиціях відповіді які відповідні спецкомандам, перетворюються і зберігаються у відповідних змінних, що приймаються (вхідних).

Якщо запит жорстко фіксований, тобто в рядку не містяться змінні дані, то рядок формується без яких-небудь команд і у такому вигляді відсилається, при цьому може бути додана контрольна сума.

Аналогічно обробляється відповідь: якщо приходить рядок, що не містить яких-небудь даних (в кінці може бути контрольна сума), це означає, що прилад працює, реагує тощо, тобто сам факт отримання відповіді від приладу вже є інформацією.

Після додавання модуля DCON (Master) [VAR] і підключення до нього універсального пристрою Universal DCON device [VAR] у конфігурації ресурсів ПЛК з'явиться незаповнений шаблон, у якого параметри і змінні не прив'язані до конкретного зовнішнього пристрою. Після заповнення полів параметрів значеннями, підключення і визначення вхідних і вихідних змінних буде сформовано конкретний пристрій для обміну даними. У вхідних змінних (*input variables*) значення записуються, коли приходить відповідь від опитуваного пристрою і в них значення фіксуються; вихідні змінні (*output variables*) використовуються при формуванні запиту.

До модуля можна додавати змінні різного розміру (8, 16 та 32 біти та 16-байтовий рядок) і типу (ціле беззнакове, з плаваючою точкою або

рядок). Тип та порядок розташування вхідних і вихідних змінних в модулі повинні відповідати рядкам команд в полях запиту і відповіді.

Отже, створіть у середовищі *CoDeSys* проект, визначіть необхідний таргет-файл та мову реалізації *POU* PLC_PRG. Як цільову платформу оберіть ПЛК *ОВЕН150-I.L*, а мову програмування оберіть *CFC*. Оскільки ПЛК лише опитуватиме модуль, то *POU* PLC_PRG залиште пустим. Далі перейдіть у вкладку «*Ресурсы*» і виберіть утиліту «*Конфигурация ПЛК*». У ній, через контекстне меню, створіть програмний модуль DCON (Master) [VAR], а в ньому замініть слот з інтерфейсом *RS-232* на *RS-485*. Далі налаштуйте параметри інтерфейсу обміну: вкажіть швидкість – *9600*, формат кадру – *8-n-1* та режим передачі – *ASCII*.

Моніторинг запитів контролера та відповідей модулів можна здійснювати за допомогою програм сканування *COM*-портів. Тобто, якщо пристрій, який підтримує протокол *DCON*, відсутній, запустіть програму сканування та налаштуйте її параметри відповідно до налаштувань ведучого пристрою в конфігурації ПЛК. Далі спостерігайте за прийнятими кадрами на екрані програми-монітора *COM*-порту. Прийняті запити розшифруйте і зіставте з налаштуваннями запитів у проекті для ПЛК. Зверніть увагу на значення контрольної суми і перевірте її правильність.

Наприклад, був прийнятий такий запит:

HEX: 40 31 30 34 30 30 35 0D або ASCII: @ 10 40 05 . .

Кадр містить два службові символи-роздільники кадрів. Це початок кадру – «40» (@) та кінець кадру – «0D» (<*CR*>, в *ASCII*-кадрі позначений крапкою). У полі даних вказана адреса пристрою – байти «31» та «30», що у *ASCII*-коді відповідає значенню «10_{hex}». Далі вказано значення маски – байти «34» та «30», що відповідає значенню «40_{hex}». У числі «40_{hex}» виставлений лише біт у п'ятому розряді, що відповідає включенню виходу № 6 модуля *МДВВ*. Байти «30» та «35» є контрольною сумою, яка відповідає значенню «05_{hex}». Для перевірки відповідності значення суми прийнятої посилки підсумуємо прийняті байти, окрім символу <*CR*>. Отже: $40_{hex}+31_{hex}+30_{hex}+34_{hex}+30_{hex}=105_{hex}$. Оскільки контрольна сума складає два символи (один байт), то залишаємо молодший байт –

05_{hex}, відповідний в запиті двом байтам «30» та «35». Остаточно можна зробити висновок: *ПЛК150* формує запит на включення виходу № 6 в модулі *МДВВ* з адресою №16.

3.5.3. Приклади запитів у протоколі DCON

1) Розглянемо приклад опитування четвертого каналу модуля *МВА* з адресою 16_{10} (10_{hex}).

Формат запиту в параметрах універсального модуля буде таким:

10 3 [+],

де # – роздільник; 10 – адреса модуля у *HEX*-форматі; 3 – номер канала (в даному випадку це четвертий канал модуля *МВА*); [+] – означає автоматичний розрахунок *CHK*.

Формат позитивної відповіді модуля буде таким:

> [7f] [+],

де **>** – роздільник, 7 – модифікатор; f – це дія, тобто тип змінної, що передається, в *ASCII*-символах в десятковому форматі із знаком, роздільником цілої та дробової частини числа (крапкою). Рядок має фіксоване число символів, задане модифікатором; [+] – означає автоматичний розрахунок *CHK*.

Вікно налаштування універсального модуля *DCON* для опитування 4-го аналогового каналу модуля *МВА* наведено на рис. 3.27. Для зберігання значення на 4-му вході *МВА* в універсальному пристрої оголошена вихідна змінна типу *float*.

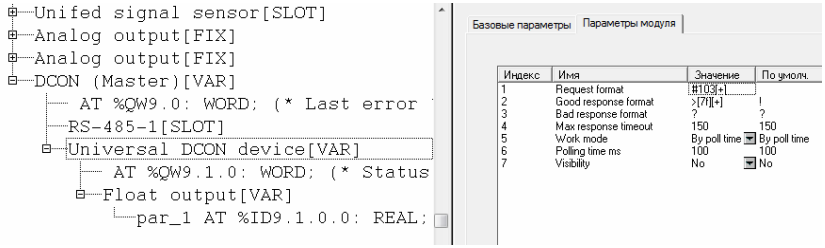


Рисунок 3.27 – Налаштування *ПЛК150* для опитування 4-го входу *МВА*

2) Розглянемо приклад опитування усіх каналів модуля *MBA* з адресою 16_{10} (10_{hex}).

Формат запиту в параметрах універсального модуля буде таким:

10 [+],

де # – роздільник; 10 – адреса модуля у *HEX*-форматі; [+] – означає автоматичний розрахунок *CHK*.

Формат позитивної відповіді модуля має вигляд:

> [7f] [7f] [7f] [7f] [7f] [7f] [7f] [7f] [+],

де **>** – роздільник, 7 – модифікатор; f – це дія, тобто тип змінної, що передається, в *ASCII-символах* в десятковому форматі із знаком, роздільником цілої та дробової частини числа (крапкою). Рядок має фіксоване число символів, задане модифікатором, причому кількість спецкоманд відповідає кількості входів модуля; [+] – означає автоматичний розрахунок *CHK*.

Вікно налаштування універсального модуля *DCON* для опитування усіх аналогових каналів модуля *MBA* показано на рис. 3.28. Для зберігання значень на входах *MBA* в універсальному пристрої оголошені вісім вихідних змінних типу *float*.

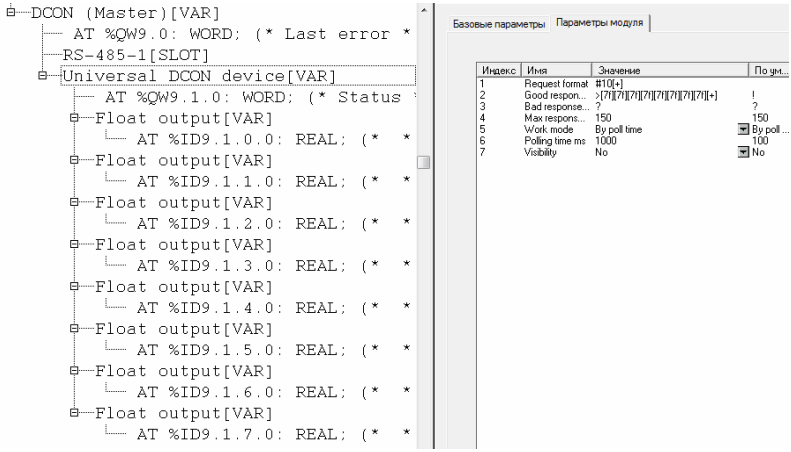


Рисунок 3.28 – Налаштування *ПЛК150* для опитування усіх входів *MBA*

3) Розглянемо приклад групового опитування усіх вхідних дискретних каналів (маски входів) модуля *МДВВ* з адресою 16_{10}.

Формат запиту в параметрах універсального модуля буде таким:

@ 10 [+],

де @ – роздільник; 10 – адреса модуля у *HEX*-формату; [+] – означає автоматичний розрахунок *CHK*.

Формат позитивної відповіді модуля має вигляд:

> [4h] [+],

де **>** – роздільник, 4 – модифікатор; h – це дія над змінною, що передається у *ASCII*-символах, у *HEX*-форматі або перетворення *ASCII*-рядка з *HEX*-формату у змінну, що приймається, причому рядок має фіксоване число символів, задане модифікатором; [+] – означає автоматичний розрахунок *CHK*.

Вікно налаштування універсального модуля *DCON* для групового опитування усіх вхідних дискретних каналів (маска входів) модуля *МДВВ* показано на рис. 3.29. Для зберігання значення маски входів модуля *МДВВ* в універсальному пристрої оголошена вихідна змінна без знакового типу розміром 16 бітів.

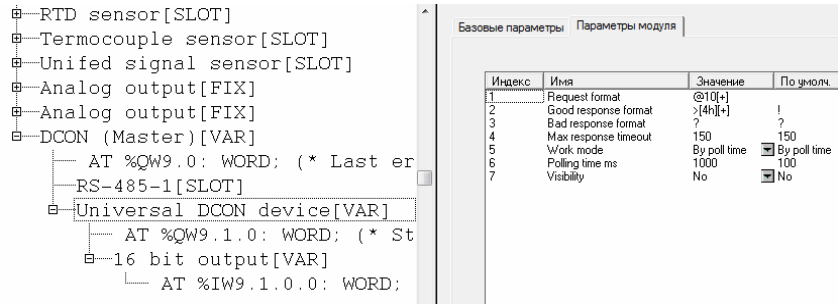


Рисунок 3.29 – Налаштування *ПЛК150* для опитування маски входів *МДВВ*

4) Розглянемо приклад запису значення групової маски виходів до модуля *МДВВ* з адресою 16_{10}.

Формат запиту на запис маски виходів у параметрах універсального модуля буде таким:

@ 10 [2h] [+],

де @ – роздільник; 10 – адреса модуля у *HEX*-форматі; [2h] – спецкоманда, що складається з модифікатора – 2, який визначає кількість символів, та дії – h, яка змінну, що передається в *ASCII*-символах, у *HEX*-форматі або перетворює *ASCII*-рядок з *HEX*-формату у змінну, що приймається; [+] – означає автоматичний розрахунок *CHK*.

Формат позитивної відповіді модуля після виконання команди запису значення маски виходів має вигляд:

> [+],

де **>** – роздільник; [+] – означає автоматичний розрахунок *CHK*.

Вікно налаштування універсального модуля *DCON* для запису значення маски дискретних виходів модуля *МДВВ* показано на рис. 3.30. Для зберігання значення маски виходів модуля *МДВВ* в універсальному пристрої оголошена вхідна змінна без знакового типу розміром 16 бітів.

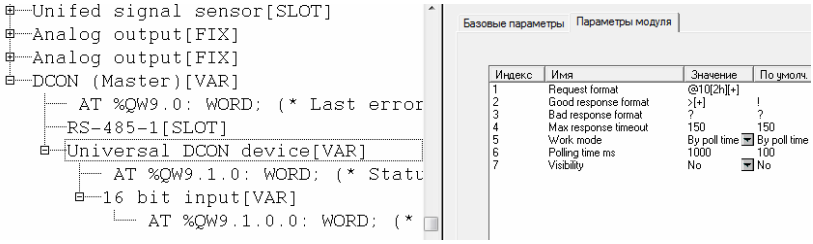


Рисунок 3.30 – Налаштування *ПЛК150* для запису маски виходів *МДВВ*

5) Розглянемо приклад читання значення лічильника на дискретному вході модуля *МДВВ* з адресою 16_{10}.

Формат запиту в параметрах універсального модуля буде таким:

@ 10 1 [+],

де @ – роздільник; 10 – адреса модуля у *HEX*-форматі; 1 – номер каналу лічильника (у даному випадку це 2-й вхід модуля МДВВ), [+] – означає автоматичний розрахунок *CHK*.

Формат позитивної відповіді модуля має вигляд:

! [5d] [+],

де **!** – роздільник, 5 – модифікатор; d – подає дію, тобто тип змінної, що передається в *ASCII*-коді в десятковому форматі (без знака) або перетворює *ASCII*-рядок із десяткового формату (без знака) у змінну, що приймається, причому рядок має фіксовану кількість символів, яка задається модифікатором; [+] – означає автоматичний розрахунок *CHK*.

Вікно налаштування універсального модуля *DCON* для опитування значення лічильника на 2-му дискретному вході модуля *МДВВ* показано на рис. 3.31. Для зберігання значення лічильника в універсальному пристрої оголошена вихідна змінна без знакового типу розміром 32 біти.

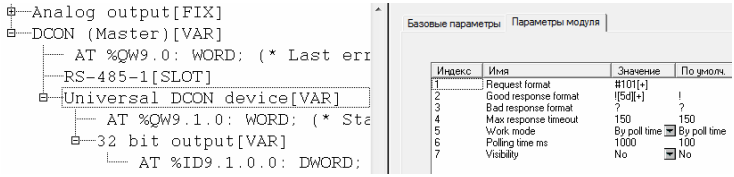


Рисунок 3.31 – Налаштування *ПЛК150* для читання лічильника на другому вході *МДВВ*

Отже, відкомпілюйте проект для опитування потрібного модуля введення/виведення (*МВА* або *МДВВ*), підключіться до ПЛК і завантажте проект до нього. Для завантаження проекту використайте інтерфейс *Ethernet*. Запустіть ПЛК на виконання програми користувача і, не відключаючись середовищем *CoDeSys* від ПЛК, спостерігайте за мережними змінними проекту та введіть значення для запису їх до модуля, наприклад для управління маскою виходів *МДВВ*, або зчитуйте значення лічильника на будь-якому вході.

Контрольні запитання:

1. Які інтерфейси використовують для реалізації протоколів *ModBus*, *OWEN* та *DCON*?
2. На яких рівнях моделі *OSI* побудовані протоколі *ModBus*, *OWEN* та *DCON*?
3. Опишить формат кадру у протоколах *ModBus*, *OWEN* та *DCON*?

4. Яким чином забезпечується достовірність даних на канальному рівні в протоколах *ModBus*, *OWEN* та *DCON*?
5. Які принципи адресації вузлів закладені у послідовних протоколах?
6. Якого типу змінні та функції описані у протоколі *ModBus*?
7. Як розділяють кадри в послідовних протоколах?
8. Яким чином ведений розпізнає запит в протоколі *OWEN*?
9. Які принципи адресації закладені в протоколі *DCON*?
10. Поясніть, у чому полягає принцип обміну даними «Ведучий пристрій» - «Ведений пристрій»?

РОЗДІЛ 4.
МЕТОДИКА РОЗРОБЛЕННЯ ЛЮДИНО-МАШИННОГО ІНТЕРФЕЙСУ НА ОСНОВІ РОЗПОДІЛЕНИХ СИСТЕМ УПРАВЛІННЯ

У цьому розділі розглянуті практичні питання щодо створення розподілених систем управління (РСУ), а також розроблення на їх основі людино-машинного інтерфейсу (ЛМІ) з використанням промислових протоколів послідовних інтерфейсів.

Так, завершивши вивчення цього розділу, ви зможете:

- самостійно розробляти РСУ та створювати ЛМІ;
- закріпити знання про специфікації протоколів *ModBus*, *OWEN* та *DCON*;
- закріпити навички щодо конфігурування ресурсів ПЛК для роботи в промисловій мережі;
- налаштовувати апаратно-програмні засоби (операторські панелі та модулі введення/виведення) РСУ, які використовують промислові протоколи послідовних інтерфейсів.

4.1. Загальна характеристика об’єкта управління та РСУ

Основним завданням цього розділу є розгляд методики створення розподіленої системи управління процесом ректифікації з людино-

машинним інтерфейсом на основі використання промислових протоколів послідовних інтерфейсів. На рис. 4.1 показана функціональна схема автоматизації ректифікаційної колони. Під час розроблення РСУ з ЛМІ на підставі аналізу отриманого технічного завдання (схеми автоматизації технологічного процесу ректифікації та запропонованих приладів і засобів автоматизації) необхідно розробити алгоритм управління колоною, структуру РСУ та прикладне програмне забезпечення (ППЗ) для ПЛК. РСУ повинна складатись із засобів управління та регулювання (ПЛК, регулятори), модулів зв'язку з об'єктом (модулі введення/виведення) та засобів *HMI* (операторська панель – ОП), які з'єднані за допомогою промислової мережі (наприклад, послідовним інтерфейсом *RS-485*). Протокол обміну в мережі обирається, виходячи зі складу апаратних засобів РСУ та їхніх можливостей.

Зазначимо, що подальший матеріал має на меті лише розгляд методики розроблення РСУ з ЛМІ для технологічної установки «ректифікаційна колона». Тому ми залишимо поза увагою саме технологічний регламент процесу ректифікації.

Як і раніше (див. розділ 3), для розробки ППЗ для контролерів *ОВЕН* необхідно використати середовище *CoDeSys*. Опис основних принципів роботи з контролерами *ОВЕН* 100-ї серії та середовищем їхнього програмування наведено в роботах [12, 13, 14, 15], а технічну підтримку можна одержати за електронною адресою http://www.owen.ua в *Internet*. Також у процесі розроблення РСУ з ЛМІ необхідно використати програми конфігурування технологічних мікропроцесорних регуляторів (*ТРМ*), пристроїв зв'язку з об'єктом (*ПЗО*) та панелей оператора (*ПО*) виробництва компанії ТОВ «ВО ОВЕН» (Україна). Як допоміжний матеріал можна використати технічну документацію на засоби автоматизації виробництва компанії «ВО ОВЕН», яку також можна одержати за електронною адресою http://www.owen.ua в *Internet*.

У готовому проекті РСУ з ЛМІ повинні бути реалізовані такі функції управління та людино-машинного інтерфейсу:

1) обмін та оброблення даних у реальному часі в ПЛК;

2) керування з боку ПЛК або ОП модулями зв'язку з об'єктом та локальними регуляторами;

3) відображення поточної інформації в операторській панелі;

4) введення журналу тривог та аварійна сигналізація і блокування;

Отже, спочатку необхідно провести аналіз функціональної схеми автоматизації (рис. 4.1) та побудувати таблицю з інформацією про вид та напрям сигналів, які формуються на виході датчиків або надходять до виконавчих механізмів з боку РСУ, а також перелік мережних змінних, якими обмінюються мережні пристрої – ПЛК, ОП та ПЗО. Як видно з рис. 4.1, функціональна схема автоматизації колони для технологічного процесу ректифікації складається з п'яти контурів регулювання (витрати початкової та готової суміші, витрати води, рівня та температури суміші в колоні) та п'яти контурів контролю (тиску, витрати та температури пари, тиску в колоні та якості суміші).

У контурах використані датчики з уніфікованим струмовим сигналом (4…20 мА). Це – дифманометри, перетворювачі тиску, термометри опору, термопари та концетратомір. Залишимо поза увагою конкретні типи та моделі датчиків, оскільки нас цікавить лише діапазон виміру фізичного параметра та тип електричних сигналів, які вони формують після перетворення неелектричних параметрів технологічного процесу (тиску, рівня, витрати тощо).

У контурах регулювання витрати початкової суміші і готового продукту (№ 1, № 7) та температури в колоні (№ 5) використані електричні пневматичні перетворювачі (*ЕПП*) та виконавчі механізми пневматичної дії (*МИМ*); в контурах регулювання рівня в колоні (№ 2) та витрати води (№ 4) – виконавчі механізми електричної дії (*МЕО* або *МЕП*).

Параметри технологічного процесу (контури регулювання та контролю) мають такі значення:

- температура в колоні – 78 °С;
- витрата початкової суміші – 12,5 $м^3$/год;
- витрата готового продукту – 10 $м^3$/год;
- витрата води – 17 $м^3$/год;
- рівень в колоні – 15 м;
- концентрація розчину – 90 %.

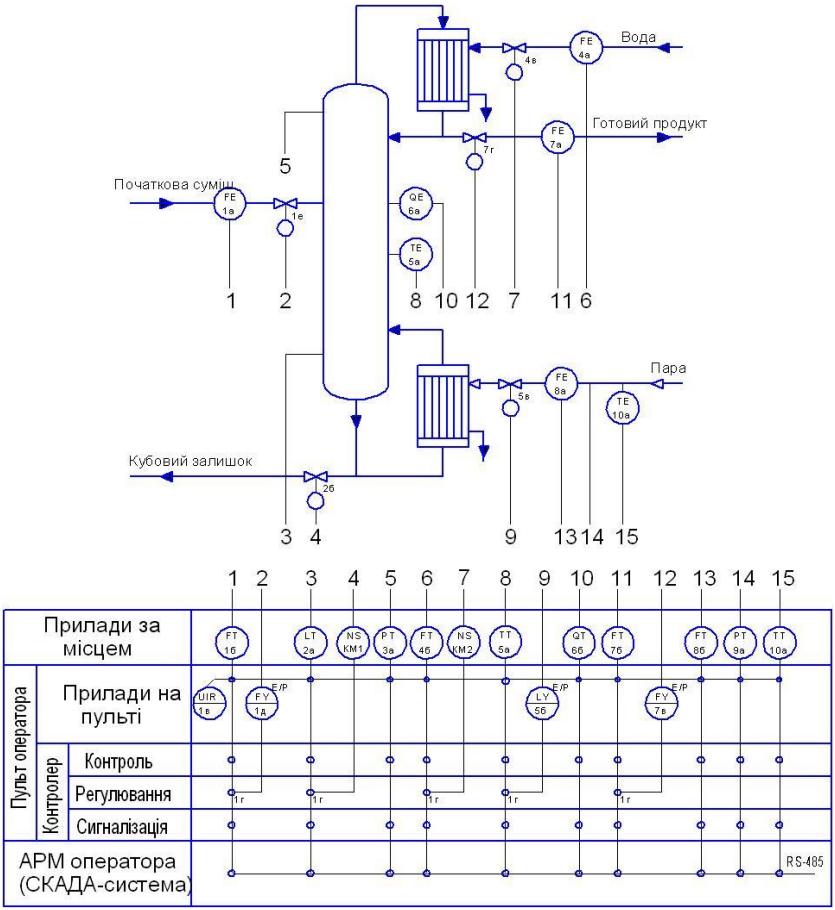


Рисунок 4.1 – Схема автоматизації ректифікаційної колони

Допустиме відхилення параметрів регулювання (гістерезис) ±5 %. У випадку більшого відхилення повинна спрацьовувати сигналізація та заноту-вуватися інформація до журналу тривог і повідомлень, а також – до звіту про хід ТП. Крім того, в контурі регулювання рівня є сигналізатори досягнення крайніх положень. Для ВМ електричної дії передбачені кінцеві вимикачі та інше додаткове устаткування для блокування та сигналізації.

У табл. 4.1 наведено відомості для контурів регулювання та управління у скороченому вигляді. Проведемо аналіз вхідних та вихідних сигналів в РСУ по контурах та зведемо отримані дані в табл. 4.2. Виходячи з даних, які наведено в табл. 4.2, оберемо відповідні засоби для системи управління та ЛМІ. Так, нехай функцію ЛМІ буде виконувати сенсорна операторська панель ОВЕН *СП-270*, а функцію регулювання – буде здійснювати контролер ОВЕН *ПЛК110-30*, у якого на борту є 18 дискретних входів (два з них – швидкісні) та 12 дискретних виходів (чотири з них – швидкісні). Проте в цьому контролері немає аналогових входів та виходів, тому для вимірювання аналогових параметрів будемо використовувати модулі аналогових сигналів: модулі введення *МВ110-224.8А* і *МВ110-224.2А* та модуль виведення сигналів *МУ110-224.8И*. Для ведення архіву можна долучити до системи модуль збору даних *МСД-200*. Нагадаємо, що розглядається приклад, який лише демонструє методику розроблення РСУ з ЛМІ та ППЗ, тому головний акцент робиться на побудові системи управління, а особливості монтажу засобів автоматизації (датчиків і виконавчих пристроїв) та їхнього підключення до приладів системи управління не розглядається.

Таблиця 4.1 – Зведені відомості зі схеми автоматизації

Номер конт.	Тип конт. (закон рег.)	Технологічні параметри	Вид сигн. ПП	Тип ВМ	Вид сигн.
1	рег. (ПИД)	витрата початкової суміші	4…20 мА	*МИМ*	4…20 мА
2	рег. (ПД)	рівень в колоні	4…20 мА	*МЕО*	*ШИМ*
3	контроль	тиск в колоні	4…20м А	-	-
4	рег. (ПД)	витрата води	4…20 мА	*МЕО*	*ШИМ*
5	рег. (ПИД)	температура в колоні	опір НСХ *ТСП100*	*МИМ*	4…20 мА
6	контроль	якість суміші	4…20 мА	-	-
7	рег. (ПИД)	витрата готового продукту	4…20 мА	*МИМ*	4…20 мА
8	контроль	витрата пари	4…20 мА	-	-
9	контроль	тиск пари	4…20 мА	-	-
10	контроль	температура пари	опір НСХ *ТСП100*	-	-

Таблиця 4.2 – Аналіз вхідних та вихідних сигналів в РСУ

Номер контура	Вхідні сигнали		Вихідні сигнали	
	дискретні	аналогові	дискретні	аналогові
1		1		1
2	4	1	2	
3		1	-	-
4	2	1	2	
5		1		1
6		1	-	-
7		1		1
8		1	-	-
9		1	-	-
10		1	-	-
Усього	6	10	4	3

Таким чином, до складу РСУ з ЛМІ входять ПЛК, ОП та додаткові модулі. Побудуємо мережну структуру РСУ з ЛМІ. Оскільки в ПЛК110-30 в наявності є по два послідовних інтерфейси *RS-232* та *RS-485*, створимо три мережні ланцюги, один на інтерфейсі *RS-232* для зв'язку ПЛК та ОП, другий – на інтерфейсі *RS-485* для зв'язку ПЛК з модулями розширення входів та виходів, а третій – на інтерфейсі *RS-485* для зв'язку ПЛК з модулем збору даних. В першій мережі обміном буде керувати ОП, в другій – ПЛК, а в третій – модуль збору даних, який теж може виконувати функцію головного пристрою в мережі на його другому інтерфейсі *RS-485* «*Приборы*» (перший інтерфейс *RS-485* «*ПК*» модуля призначений для зв'язку з ПК, причому лише ПК може бути головним пристроєм. Така побудова мережі обрана для того, щоб знизити загальне завантаження контролера, на якого покладається основна функція – керування технологічним процесом. В усіх ланцюгах буде застосований протокол обміну *ModBus-RTU*, причому параметри інтерфейсу обираються, виходячи з результатів тестування системи (необхідно враховувати довжину ліній зв'язку для вибору швидкості передавання даних та час затримання сигналів).

4.2. Розроблення структурної схеми РСУ з ЛМІ

Отже, виходячи з функціональної схеми автоматизації технологічного об'єкта (ТО) – ректифікаційної колони (рис. 4.1), побудуємо загальну структурну схему РСУ технологічним об'єктом, яка зображена на рис. 4.2. Ця схема складається із засобів автоматизації (датчики та виконавчі механізми), контролера (ПЛК), модулів (ПЗО) та людино-машинного інтерфейсу (ОП).

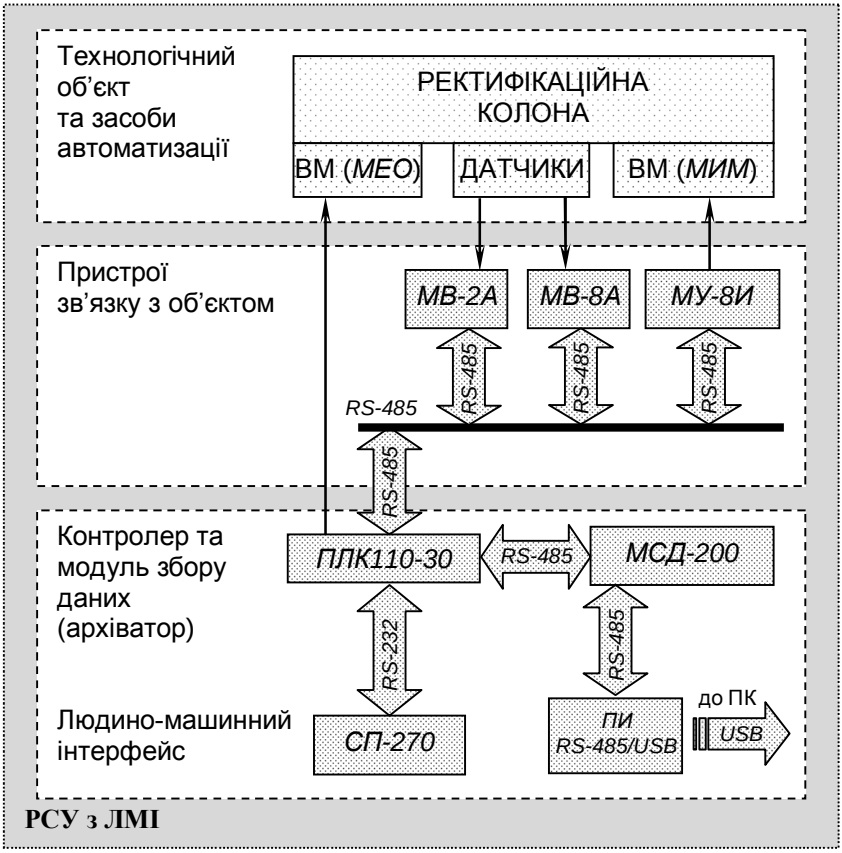


Рисунок 4.2 – Загальна структурна схема РСУ з ЛМІ

Надамо короткий опис розробленої структурної схеми (рис. 4.2). Побудована РСУ розділена на два ієрархічні рівні. Це рівень датчиків та ВМ (технологічний рівень) та рівень управління процесом (ПЛК та модулі). До верхнього рівня також входить ЛМІ, який представлений операторською панеллю. Сигнали 4…20 мА від датчиків надходять до аналогових входів модулів *МВ110-2А* та *МВ110-8А*. Контролер реалізує алгоритми регулювання з передаванням управляючого аналогового сигналу 4…20 мА через модуль виведення *МВ110-8И* на ВМ типу *МИМ* за допомогою *ЕПП* (контури № 1, 5, 7), який на схемі не показаний. Контролер також безпосередньо генерує сигнали «більше» та «менше» для управління обмотками електродвигуна ВМ типу *МЕО* (контури № 2, 4). Приклади проектів алгоритмів регулювання наведені в [4], причому для контурів № 1, № 5 та № 7 необхідно використовувати ПІД-регулятор з аналоговим вихідним сигналом 4–20 мА з його подальшим перетворенням у пневмосигнал для впливу на *МИМ*, а для контурів № 2 та № 4 – ПД-регулятор з двома імпульсними вихідними сигналами «більше» та «менше» для управління ВМ *МЕО*. Операторська панель *СП-270* слугує для відображення значення поточних параметрів у контурах регулювання та введення значень уставок у програмні регулятори. Крім того, в панелі реалізовано контроль за технологічною установкою в цілому. На схемі (див. рис. 4.2) показані усі інформаційні зв'язки пристроїв, які входять до складу РСУ з ЛМІ. Показано також зв'язок модуля збору даних з ПК, до якого буде періодично відправлятись архів за даними технологічного процесу.

Для програмування ПЛК та конфігурування інших приладів РСУ використовується звичайний ПК в стандартній комплектації з операційною системою *MicroSoft Windows XP/Vista/7* та встановленим спеціальним програмним забезпеченням (СПЗ). На ПК встановлені середовище для програмування контролерів *ОВЕН* – це *CoDeSys V2.3* та допоміжні програми *«Конфигуратор М110»*, *«Конфигуратор МСД»* та «*Конфигуратор СП200*». Підключення програм-конфігураторів до модулів введення-виведення здійснюється за допомогою ПІ *RS-232/RS-485*. Конфігурування *МСД200* здійснюється через послідовний інтерфейс *USB*.

4.3. Розроблення структурної та функціональної схем ППЗ РСУ з ЛМІ

Для розроблення структурної схеми ППЗ РСУ необхідно визначити потрібні програмні та апаратні засоби, які будуть використані при створенні проекту РСУ з ЛМІ. Так, для прикладу, що розглядається, була розроблена структурна схема ППЗ РСУ, яка зображена на рис. 4.3. На цій схемі показані програмні засоби, які необхідно використати для створення ППЗ РСУ. Структурно до складу ППЗ РСУ входять ППЗ контролера та ППЗ людино-машинного інтерфейсу. Крім того, для налаштування модулів введення-виведення та модуля збору даних застосовуються програми-конфігуратори. За допомогою цих програми налаштовуються аналогові входи та виходи пристроїв та визначаються параметри зв'язку з ПЛК (формат кадру, швидкість передавання кадрів та протокол обміну).

Розроблене ППЗ ПЛК та ЛМІ також складається з елементів, тобто підсистем. До складу ППЗ ПЛК входить проект з програмними компонентами: основна програма *PLC_PRG* з функціональними блоками (ФБ) для регуляторів температури в колоні, витрат (води, початкової та готової суміші) та рівня в колоні. В проекті також передбачені функціональні блоки для обміну даними, сутність яких полягає у пересиланні значень параметрів з одного програмного мережного модуля до іншого для подальшого обміну в мережі між пристроями. Підсистему ППЗ ЛМІ, яка розробляється в програмі конфігурування панелі оператора, складають підсистеми: інформаційного забезпечення оператора, відображення технологічних параметрів, автоматизованого управління ТО.

У результаті аналізу структурної схеми ППЗ РСУ, яка показана на рис. 4.3, отримаємо функціональну схему ППЗ РСУ, що наведена на рис. 4.4. На схемі є скорочення: ПАЗ – це протиаварійний захист, а БРУ – блок ручного управління.

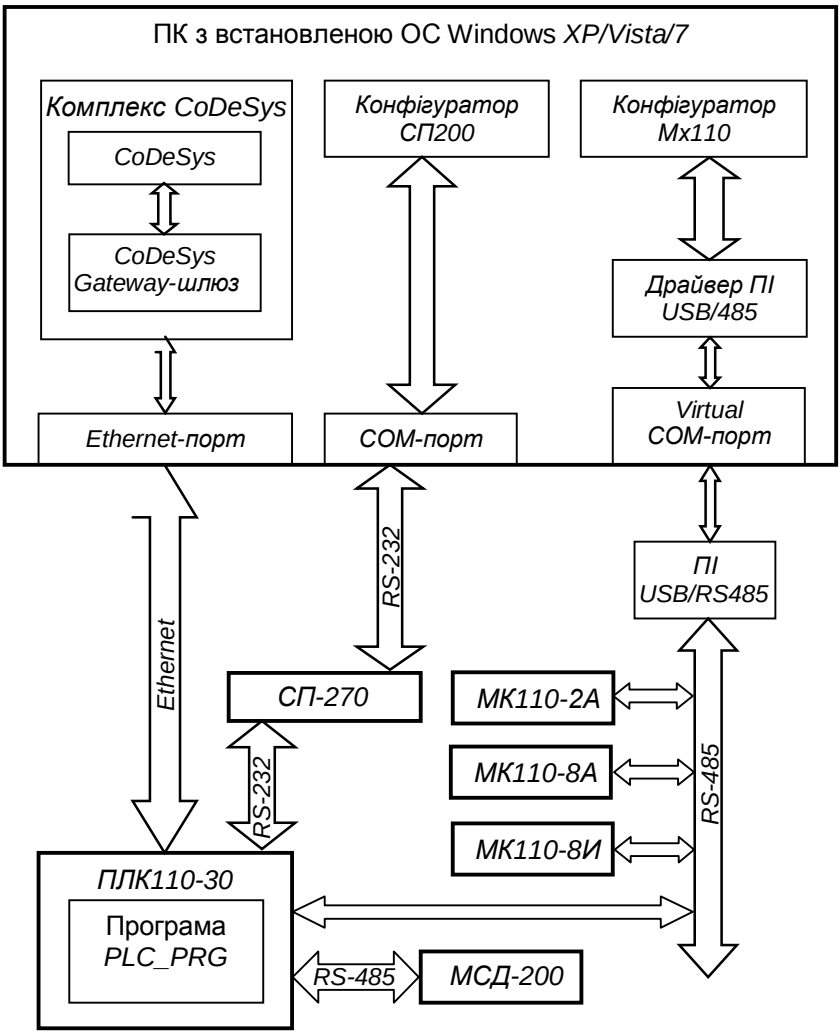


Рисунок 4.3 – Структурна схема ППЗ РСУ

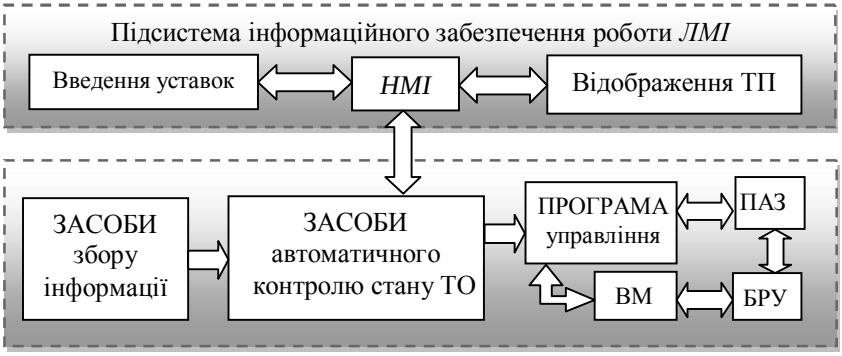


Рисунок 4.4 – Функціональна схема ППЗ РСУ

4.4. Розроблення ППЗ для *ПЛК1хх ОВЕН*

Для розроблення ППЗ контролера необхідно використати так званий технологічний регламент, а саме основні технологічні параметри та алгоритм роботи технологічної установки. Процес ректифікації протікає безперервно, тому проект з управляючою програмою можна розробити у вигляді основного програмного організаційного компонента з ім'ям *PLC_PRG*, а програми регуляторів будуть викликатися з основної програми у вигляді ФБ. Конфігурування ресурсів ПЛК (фізичних входів та виходів) необхідно провести відповідно до функціональної схеми автоматизації. В основній програмі *PLC_PRG* потрібно передбачити різноманітні режими роботи установки, сигналізацію та функціональні вимоги для забезпечення нормальної роботи технологічного об'єкта, а також ФБ для інформаційного обміну з панеллю оператора, модулями аналогових входів та виходів та модулем архівування.

Для одного і того ж самого технологічного об'єкта управління можуть бути запропоновані різні експлуатаційні (технологічні) режими: запуск та зупинка, забезпечення умов нормального функціонування та реагування на виникнення різноманітних нештатних та аварійних ситуацій. Вимоги до функціонування об'єкта можуть бути задані у вигляді технологічних, режимних або рецептурних карт, а також функціональних циклограм.

Здійснення ПІД-регулювання для управління технологічними процесами в середовищі *CoDeSys* реалізується за допомогою додаткових бібліотек. Це, наприклад, бібліотека *UTIL.lib* з папкою *Controller*, яка входить до інсталяційного пакета *CoDeSys*. До цієї папки входить три функціональних блоки: ПД- , ПІД- регулятори та ПІД-регулятор з фіксованим часом циклу. Крім того, розробники ПЛК на платформі *CoDeSys* пропонують власні бібліотеки для своїх ПЛК. Так, компанія ТОВ «ВО ОВЕН» для реалізації ПІД-регулювання та управління різними виконавчими механізмами надає додаткові безкоштовні бібліотеки *PID_Rerulators.lib* та *pid_reg2.lib*. Але особливістю цих бібліотек є неможливість виконання ФБ бібліотеки в емуляторі ПЛК (це так звані «внутрішні бібліотеки»), тобто для налагодження програм користувача потрібна обов'язкова наявність реального ПЛК. Крім того, ці бібліотеки не працюють в ПЛК інших виробників. Нагадаємо, що основне завдання ФБ з цих бібліотек – це підтримання поточного параметра (*PV*) на заданому рівні (*SP*) шляхом впливу на виконавчі механізми різної дії (з аналоговим та ШІМ-управлінням).

Отже, почнемо розробляти ППЗ для ПЛК. Насамперед, після запуску середовища програмування *CoDeSys* створіть новий проект, дайте йому ім'я, наприклад *kolona_rect.pro*. В якості цільової платформи виберіть контролер *ПЛК110-30.М ОВЕН*, а мову реалізації *POU PLC_PRG – CFC*. Додайте до проекту необхідні бібліотеки. Створіть потрібні організаційні компоненти (програми, ФБ та функції) для структурування проекту. Далі проведіть конфігурування ресурсів ПЛК відповідно до функціональної схеми автоматизації (рис. 4.1) та обраних типів датчиків і виконавчих механізмів.

4.4.1. Розроблення функціональних блоків для регулювання

На рис. 4.5 показаний фрагмент програми з ПІД-регулятором, функціональний блок якого входить до бібліотеки *UTIL.lib*. У цьому прикладі елементи з номерами *0* та *1* здійснюють періодичний перезапуск ПІД-регулятора для усунення накопичення статичної помилки регулювання. Послідовності елементів з номерами *4–5–6–7* та *8–9–10–11* забезпечують перетворення управляючого сигналу, який розрахований в ПІД-регуляторі, в значення ШІМ-сигналу на дискретному вході для забезпе-

чення роботи установки в режимах «нагрівач» або «охолоджувач» відповідно. При цьому спочатку відкидаються негативні значення сигналу (*4–5–6–7*), потім здійснюється його масштабування з метою лінійного приведення сигналу діапазону (0…100 %) до діапазону ШІМ-регулятора (0…65535 одиниць) і, насамкінець, перетворення змінної типу *REAL* на тип *WORD*. Аналогічно працює послідовність *8–9–10–11*, але в протилежному розумінні, тобто відкидаються позитивні значення сигналу. Далі перетворення аналогічні наведеному вище. Звісно, якщо в установці немає якогось одного з виконавчих пристроїв (нагрівача або охолоджувача), то в програмі залишається відповідна послідовність для управління наявним ВМ. Також ця програма може бути використана для управління електричними виконавчими механізмами типу *МЕО* або *МЕП*.

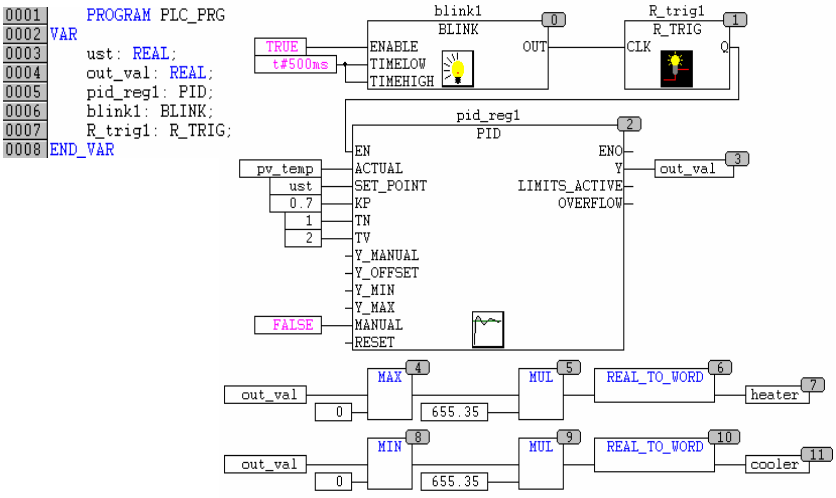


Рисунок 4.5 – Фрагмент програми з ПІД-регулятором з дискретними виконавчими механізмами

Наступний приклад демонструє порядок використання ФБ з бібліотеки *PID_Rerulators.lib*, яка розроблена спеціально для контролерів ОВЕН. В даному випадку ПІД-регулятор аналогічний попередньому прикладу, але до нього добавлена функція автоналаштування його параметрів до властивостей об’єкта. На рис. 4.6 зображений фрагмент програми

користувача для ПІД-регулювання за допомогою 2-позиційного ВМ та можливістю вмикання та вимикання режиму автоналаштування ПІД-регулятора. Також в проекті передбачений ФБ *DIG_FLTR*, який зменшує вплив випадкових імпульсних перешкод на вимірюваний параметр. В розділі оголошення змінних перші чотири змінних різних типів зв'язані з відповідними фізичними входами та виходами ПЛК. Інші змінні є локальними і потрібні для функціонування програми користувача та контролю за режимом роботи ПІД-регулятора.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     vh1_znach AT%ID4.0 : REAL;     (*Измеренное значение на аналоговом входе №1*)
0004     vh1_vrem AT%IW4.1 : WORD;      (*Измеренное время нааналоговом  входе №1*)
0005     zapusk_anr AT%IX0.0 :BOOL;     (*Дистанционный запуск АНР с помощью кнопки на дискретном входе №1*)
0006     vih1 AT%QW1.0.0 :WORD;         (*к дискретному выходу №1 подключен подмодуль ШИМ*)
0007     filtr1: DIG_FLTR;                  (*Цифровой фильтр на измеренное значение*)
0008     ust: REAL := 50;                   (*Уставка, с предустановкой 50*)
0009     anr_start: BOOL := TRUE;       (*Команда запуска автонастройки. При первом запуске включается автоматически*)
0010     status_anr: BYTE;                  (*Статус прохождения автонастройки*)
0011     triger: SR;
0012     promegut: BOOL;
0013     promegut1: BOOL;
0014     trig1: R_TRIG;
0015     pid1: PID_2POS_IM_ANR;         (*Блок ПИД с автонастройкой*)
0016 END_VAR
```

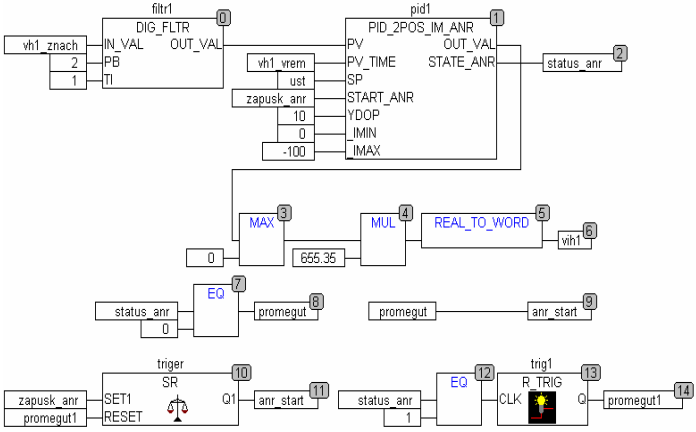


Рисунок 4.6 – Фрагмент програми з ПІД-регулятором з автоналаштуванням

На відміну від попереднього прикладу, де реалізовано ПІД-управління 2-позиційним ВМ, у наступному прикладі буде розглянута

можливість ПІД-управління 3-позиційним ВМ з регулювальним органом типу «клапан» та автоналаштуванням до властивостей об'єкта регулювання. В цьому випадку буде застосований ФБ із бібліотеки *pid_reg2.lib*. Відзначимо, що для реалізації режимів управління ВМ «більше» та «менше» в проекті використано два перших дискретних виходи ПЛК, причому сигнальні лінії підключені до нормально закритих контактів перемикаючих входів ВМ. Фрагмент програми користувача, яка використовує ФБ *APID_VALVE* з бібліотеки *pid_reg2.lib*, показаний на рис. 4.7. В розділі оголошення змінних перші п'ять змінних різних типів зв'язані з відповідними фізичними входами та виходами ПЛК. Інші змінні є локальними і потрібні для функціонування програми користувача та контролю за режимом роботи ПІД-регулятора.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     vh_znach AT%ID4.0 : REAL;     (*Измеренное значение на аналоговом входе №1*)
0004     vh_vrem AT%IW4.1 : WORD;      (*Измеренное время на аналоговом  входе №1*)
0005     anr_start AT%IX0.0 :BOOL;  (*Дистанционный запуск АНР с помощью кнопки на дискретном входе №1*)
0006     vih1 AT%QX1.0 :BOOL;          (*к дискретному выходу №1*)
0007     vih2 AT%QX1.1 :BOOL;          (*к дискретному выходу №2*)
0008     filtr1: DIG_FLTR;             (*Цифровой фильтр на измеренное значение*)
0009     ust: REAL := 40;              (*Уставка, с предустановкой 40*)
0010     vrem_hod: REAL := 90;         (*время хода задвижки, в зависимости от характеристик ИМ*)
0011     reg_1: APID_VALVE;            (*ПИД регулятор с автонастройкой для трехходовых ИМ без датчика положения*)
0012     nach_moshn: REAL := 20;       (*Начальное значение мощности при автонастройке*)
0013     zona_nechustv: REAL := 5;     (*Зона нечуствительности в ед.параметра*)
0014     luft: REAL := 0.5;            (*Время выборки люфта при реверсе, в секундах*)
0015     sinhro: BOOL;                 (*Задавать положение задвижки из мат. модели *)
0016     moshn_upr: REAL;
0017     pr_anr: BOOL;
0018     pr_bv_ust: BOOL;
0019 END_VAR
```

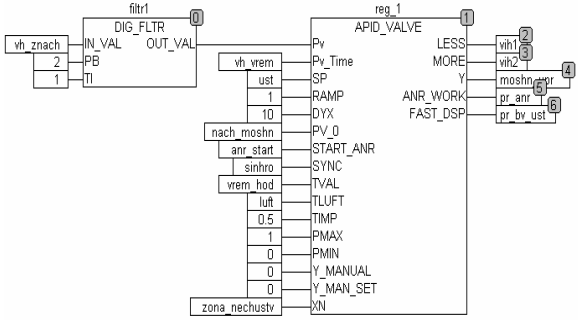


Рисунок 4.7 – Фрагмент програми з ПІД-регулятором з автоналаштуванням для управління клапаном

На рис. 4.8 наведений приклад ПІД-регулятора на базі ФБ з бібліотеки *UTIL.lib* для управління трипозиційним ВМ без датчика положення (ДП). Для управління клапаном з електричним ВМ типу *MEO* використаний ФБ *VALVE_REG_NO_POS* з бібліотеки *pid_reg2.lib*. Виходи *LESS* та *MORE* цього блока підключені до обмоток електродвигуна для його обертання в протилежні сторони, що приводить до відкриття або закриття клапана.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     filtr1: DIG_FLTR;          (*Блок фильтрации измеренного значения*)
0004     t1: REAL := 20;            (*Измеренная температура*)
0005     polosa: REAL;              (*Полоса фильтра, в единицах измеренной величины*)
0006     vrem: REAL := 2000;        (*Полоса времени, в мс*)
0007     pid1: PID_FUNCTION;        (*ПИД регулятор на поддержание*)
0008     t_izm: WORD;               (*Время измерений, берется из аналогового входа, либо тактируется в программе*)
0009     ust: REAL := 15;           (*Значение уставки, которое необходимо поддерживать*)
0010     klapan1: VALVE_REG_NO_POS; (*Блок управления импульсным клапаном больше\меньше*)
0011     out1: BOOL;                (*Сигнал о закрытии заслонки - на второй выход*)
0012     out2: BOOL;                (*Сигнал о закрытии заслонки - на первый выход*)
0013 END_VAR
```

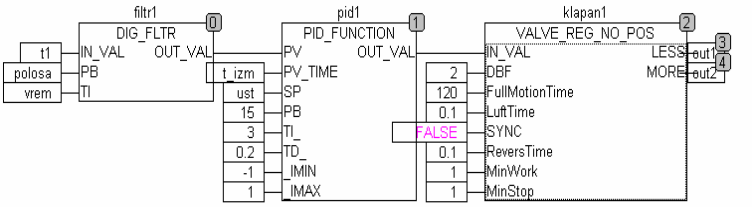


Рисунок 4.8 – Фрагмент програми з ПІД-регулятором та ФБ для управління клапаном без датчика положення

Насамкінець, наведемо приклад ПІД-регулятора для управління ВМ типу *МІМ*. У цьому проекті вихідний уніфікований струмовий сигнал (УСС) 4…20 мА після ФБ *PID* надходить до аналогового виходу ПЛК, а далі перетворюється в пневматичний сигнал, який впливає на діафрагму ВМ, яка з'єднана з клапаном за допомогою штока. Якщо на виході ПЛК значення сигналу дорівнює 4 мА, то це означає, що клапан повністю закритий. Навпаки, клапан повністю відкритий, якщо значення сигналу дорівнює 20 мА. Відповідно, якщо клапан знаходиться в положенні 50 %, тоді УСС дорівнює 12 мА. Фрагмент програми з таким регулятором показано на рис. 4.9.

Отже, було наведено приклади використання бібліотечних компонентів для реалізації ПІД-регулювання за допомогою ВМ різних типів. Зауважимо, що при розробці ППЗ для ПЛК необхідно враховувати експлуатаційні характеристики та принцип дії ВМ.

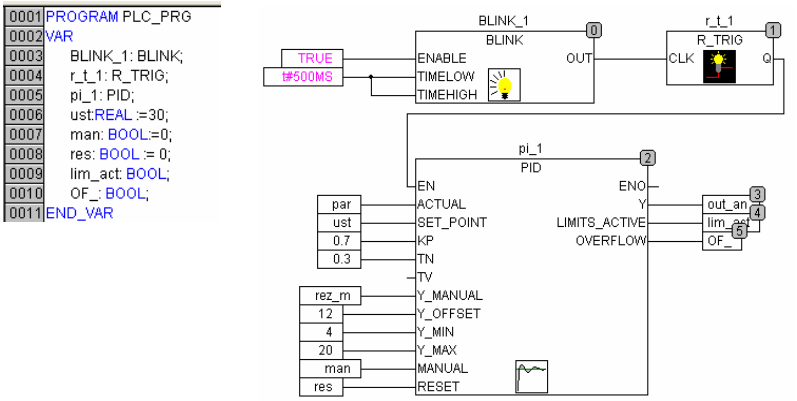


Рисунок 4.9 – Фрагмент програми з ПІД-регулятором
та ФБ для аналогового управління клапаном

4.4.2. Розроблення структури мережного обміну в конфігурації ресурсів ПЛК

Враховуючи, що ПЛК є центральним пристроєм системи управління, на якого покладені обов'язки реалізації функцій регулятора, функцій збору даних від модулів введення аналогових сигналів, функцій управління модулями виведення аналогових сигналів та функцій підтримання зв'язку з операторською панеллю та модулем збору даних, окремо розглянемо порядок створення програмних мережних модулів в конфігурації його ресурсів. Також для одного контуру регулювання покажемо увесь ланцюг проходження даних від модуля до операторської панелі та модуля збору даних.

Нагадаємо, що при розробленні РСУ з ЧМІ на послідовних інтерфейсах з використанням протоколу *ModBus* обов'язково потрібен головний пристрій. Зазвичай, для розвантаження ресурсів ПЛК функцію головного пристрою покладають на ОП. Але, коли застосовують ПЗО, які

не можуть бути ведучим пристроєм, то по іншому інтерфейсу ПЛК повинен виконувати функцію головного пристрою. Для архівування даних в РСУ передбачено використання модуля збору даних, який може виконувати функцію головного пристрою. Тому до конфігурації ресурсів ПЛК необхідно додати три програмних модулі для мережного обміну, один для зв'язку з ОП по інтерфейсу *RS-232*, другий – для зв'язку з ПЗО по інтерфейсу *RS-485 (1)*, а третій – для зв'язку з модулем збору даних по другому інтерфейсу *RS-485 (2)*. Відмітимо, що в *ПЛК110-30* є в наявності по два інтерфейси *RS-232* та *RS-485*. В усіх програмних модулях як протокол обміну буде використаний *ModBus-RTU*. Порядок налаштування мережних модулів в ПЛК розглянуто в третьому розділі. Тому буде розглянуто лише налаштування операторської панелі *СП-270* та модулів зв'язку з об'єктом і модуля архівування даних *МСД200*.

У створеному проекті з функціональними блоками регуляторів необхідно перейти до вкладки, де здійснюється конфігурування ресурсів ПЛК, а саме до вкладки «*Ресурсы*», та обрати категорію «*Конфигурация ПЛК*». Для зв'язку з ОП в ПЛК додайте до конфігурації ресурсів ПЛК та налаштуйте програмний мережний модуль ModBus (slave) [VAR] (див. пункт 3.3.2). Додайте до модуля інтерфейс обміну (RS-232 [VAR]) та налаштуйте його параметри (115200, 8-n-1). Аналогічно додайте модуль ModBus (slave) [VAR] для зв'язку з модулем архівування даних, але інтерфейс обміну додайте RS-485-1 [VAR]. Насамкінець, додайте програмний модуль головного пристрою ModBus (Master) [VAR] з універсальними підпорядкованими модулями Universal Modbus device [VAR] для обміну даними з модулями вхідних та вихідних аналогових сигналів по інтерфейсу RS-485-2 [VAR] (див. пункт 3.3.1). В результаті отримаєте структуру ресурсів ПЛК для мережного обміну по трьох інтерфейсах, як це показано на рис. 4.10.

4.5. Конфігурування пристроїв, які входять до РСУ з ЛМІ

Усі пристрої, які входять до РСУ з ЛМІ, окрім контролера, потребують налаштування конфігураційних параметрів. Це, по-перше, налаштування для мережного обміну, а по-друге, це конфігурування ресурсів

пристрою. Тому розглянемо порядок конфігурування пристроїв, які входять до РСУ з ЛМІ, а саме панелі оператора *СП-270*, модуля архівування даних *МСД200* і пристроїв зв'язку з об'єктом *МВ110-8А*, *МВ110-2А* та *МУ110-8И*. Для налаштування перелічених пристроїв використовують програми-конфігуратори, які додаються до пристроїв їхнім виробником, компанією ТОВ «ВО ОВЕН».

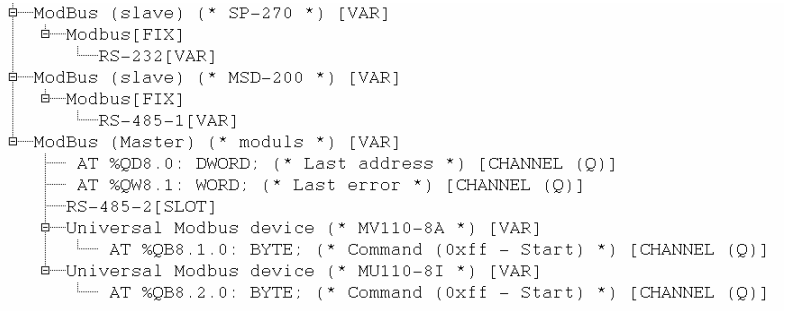


Рисунок 4.10 – Конфігурація ресурсів ПЛК для мережного обміну

4.5.1. Налаштування обміну даними між ПЛК110-30 та модулями МВ110-8А, МВ110-8А і *МВ110-8И*

Доступ до даних ПЗО, тобто до локальних регуляторів серії *ТРМ* та модулів введення-виведення (*МВА*, *МДВВ*, *Мх110* тощо), здійснюється з боку ПЛК по послідовному інтерфейсу *RS-485* з використанням протоколів *ModBus*, *OWEN* або *DCON* [18, 19]. Але це можливо, якщо в мережі є ведучий пристрій. Це може бути, наприклад ПЛК або ПО. Для цього до конфігурації ресурсів ПЛК необхідно додати програмні мережні модулі ModBus (Master) [VAR], OWEN (Master) [VAR] або DCON (Master) [VAR].

Якщо додати програмний мережний модуль ModBus (Master) [VAR], то порядок його налаштування аналогічний розглянутому у пункті 3.3.1. При створенні віртуального пристрою *Universal Modbus Device* в конфігурації ресурсів модуля ModBus (Master) [VAR] є можливість використання готових шаблонів модулів *МВА8* та *МДВВ*. Зауважимо, що перед використанням модуля потрібно налаштувати протокол обміну та параметри зв'язку за допомогою програм для конфігурування відповідних пристроїв.

У програмному модулі головного пристрою для зв'язку з модулем аналогових входів створимо та налаштуємо мережні змінні для одного контуру, наприклад контуру регулювання рівня в колоні (контур № 2). Нагадаємо, що в контурі застосовується ВМ електричної дії (*МЕО*). На рис. 4.7 в переліку змінних регульованим параметром є змінна з ім'ям vh_znach типу *REAL*, значення уставки зберігає змінна ust типу *REAL*. Інші змінні є додатковими. Ці змінні забезпечують роботу ФБ ПІД-регулятора з автоналаштуванням. В коментарях до переліку змінних вказано призначення усіх змінних.

У програмному модулі головного вузла (ModBus (Master) [VAR]) створимо модуль підпорядкованого вузла (Universal Modbus device [VAR]), а в ньому мережні змінні для отримання значення параметра від модуля аналогових входів та часу вимірювання параметра. Останній параметр потрібен для обчислень в ПІД-регуляторі. Параметри налаштування модуля та змінних подібні тому, як це показано в *пункті 3.3.1* на рис. 3.6 та рис. 3.7. Але замість змінних типу вхідний (Register input module [VAR]) та вихідний (Register output module [VAR]) регістри додайте змінну типу *float* (Real input module [VAR]). Адресу модуля визначіть самостійно (нехай адреса буде дорівнювати «16»), а адресу регістра зберігання значення параметра на вимірювальному вході визначіть за допомогою керівництва з експлуатації модуля *МВ110-8А*. Звертаємо увагу на те, що в десятому рядку параметрів модуля «*Byte Sequence*» (послідовність слідування байтів) потрібно значення *Trace_mode* замінити на значення *Native*. Це робиться для того, щоб правильно розшифровувати змінні, які зберігаються в двох регістрах, наприклад, змінні типу *float* (див. стор. 118). Далі значення технологічного параметра обробляється у відповідному екземплярі ФБ (див. рис. 4.7). Керування обмотками електродвигуна ВМ типу *МЕО* здійснюється за допомогою дискретних виходів на борту ПЛК.

4.5.2. Налаштування обміну даними між ПЛК110-30 та панеллю оператора СП-270

Для організації мережної взаємодії ПЛК та ПО в конфігурації ресурсів ПЛК був створений програмний модуль підпорядкованого при-

строю з інтерфейсом зв'язку *RS-232* (рис. 4.10). Зараз за допомогою програми-конфігуратора потрібно провести конфігурування операторської панелі *СП-270*. Для налаштування конфігураційних параметрів панелі потрібно використати інформацію з [22]. Запустіть програму «*Конфигуратор СП200*» та встановіть зв'язок програми з панеллю оператора. Ця програма призначена для конфігурування панелі оператора *СП-270*. Програма працює під управлінням ОС *MS Windows 98/2000/XP* та дозволяє формувати і зберігати призначені для користувача екрани, що відображаються на дисплеї панелі в процесі експлуатації, та налаштовувати обмін між ПО і ПЛК даними, які відображаються на екранах користувача. Нагадаємо, що конфігурування параметрів ПО і зв'язок з ПЛК здійснюється по інтерфейсу *RS-232* (використовується інтерфейсний порт *Download*). Робота з програмою конфігурування ПО полягає в створенні екранів, виборі необхідних графічних елементів та налаштуванні їх властивостей. Для конфігурування графічних елементів на екранах панелі оператора *СП-270* використовуйте *Руководство пользователя* [22]. Після закінчення створення проекту збережіть його та завантажте його до панелі, використовуючи стандартні процедури меню програми. Конфігурація панелі є сукупністю значень параметрів, які визначають її роботу. Проект з конфігурацією панелі зберігається в файлі з розширенням *.twp .

Панель оператора, яка є в ЛМІ у складі РСУ, виконує такі функцій:

- відображення за допомогою графічних елементів стану керованого об'єкта в режимі реального часу;
- відображення сенсорних елементів, за допомогою яких оператор здійснює безпосереднє управління функціонуванням ТО;
- управління функціонуванням ПЛК і/або іншими пристроями РСУ;
- запис та читання значень регістрів ПЛК і/або інших пристроїв, з якими з'єднана ПО.

Для забезпечення роботи ПІД-регулятора в контролері операторська панель виконує перелічені вище функції за допомогою таких змінних:

- відображення поточного значення параметра;
- ручне введення уставки;
- вмикання та вимикання режиму автоналаштування;

- перехід на ручний або автоматичний режим керування ВМ;
- зона нечутливості регулятора;
- статус ПІД-регулятора;
- вихідна потужність регулятора;
- сигналізація режиму роботи ВМ – «більше» або «менше».

Виходячи з переліку мережних змінних, які потрібні для роботи ПІД-регулятора, в програмному модулі ресурсів ПЛК веденого пристрою створимо необхідні змінні. На рис. 4.11 наведено приклад структури веденого пристрою в ПЛК для обміну даними з ПО. Перші чотири змінні типу 8 bits [VAR] призначені для обміну даними типу *BOOL*. В переліку input_0 вхідні змінні керують режимом автоналаштування та ручним керуванням ВМ. Перелік output_0 вміщує змінні, які відображують стан ВМ та ПІД-регулятора. Можливе також відображення досягнення крайніх положень ВМ. Також в структуру веденого пристрою входять п'ять змінних типу *REAL*. Це значення поточного параметра, уставки тощо.

Отже, в програмі конфігурування панелі створіть новий проект з режимом роботи панелі оператора, яка буде виконувати функцію головного пристрою. У програмі передбачено покрокове конфігурування знов створеного проекту за допомогою «*Мастера настройки*» з діалоговими вікнами. Так, на першому діалоговому вікні виберіть тип панелі – «*СП-270*» (поки що це єдиний тип панелі), на другому виберіть призначення порту з ім'ям «*PLC*» *(*це буде протокол «*Modbus RTU* (*Панель Мастер*)») та налаштуйте параметри інтерфейсу, на третьому вікні – виберіть призначення порту з ім'ям «*Download*»*:* «*Modbus RTU* (*Панель Мастер*)», далі, в наступному вікні введіть ім'я проекту та натисніть на кнопку «*Готово*». В результаті з'явиться стандартне дерево проекту з одним пустим екраном.

Якщо в подальшому буде потрібно змінити налаштування інтерфейсу обміну з ПЛК, насамперед, за допомогою стандартних процедур, необхідно зберегти проект на жорсткий диск. Далі в меню «*Файл*» вибрати команду «*Настройка*…» і в діалоговому вікні, що відкрилося, вибрати вкладку «*Устройства*». У даній вкладці поставити покажчик напроти поля «*Ведомый*», а в полі «*Порт Download* – *Modbus RTU* (*Панель Мас-*

тер) і налаштувати параметри інтерфейсу, натиснувши на кнопку «…» напроти тексту «*Параметры» (*за умовчанням встановлено такі параметри – *9600, 8-е-1*). У полі напроти тексту «*Адрес устройства*» вкажіть необхідну адресу, нехай це буде «16». Після того як необхідні значення визначені – натисніть кнопку «*ОК*». Вікно закриється, а введені дані будуть збережені в проекті. Тепер все готово для створення людино-машинного інтерфейсу (тобто екранів панелі).

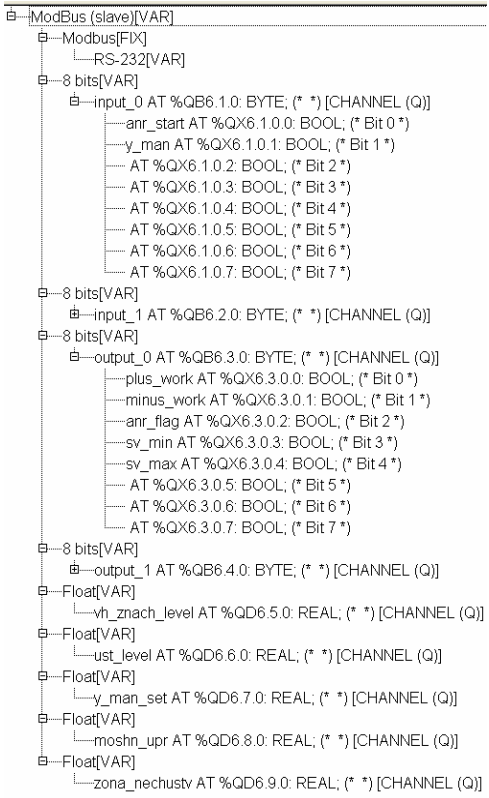


Рисунок 4.11 – Структура веденого пристрою в конфігурації ресурсів ПЛК для мережного обміну з ПО

На рис. 4.12, показаний варіант оформлення одного з екранів панелі для реалізації контуру регулювання рівня в колоні (контур № 2 на рис. 4.1).

На мнемосхемі зображені графічні елементи та об'єкти різного функціонального призначення:

- статичний текст, не пов'язаний з мережними елементами – це назва екрана – «*КОЛОННА*»;
- динамічний текст, залежить від значення змінної;
- поля для відображення поточних параметрів технологічного процеса – витрата почасової сировини та готового продукта, рівень в колоні, прочитані в ПЛК;
- поля для введення значень уставок для запису в ПЛК;
- тумблери встановлення режимів роботи роботи виконавчих механізмів для запису в ПЛК;
- стрілочні індикатори для відображення значень технологічних параметрів, прочитані в ПЛК;
- імітатори роботи установки – інтерактивні трубопроводи і насоси, залежні від значення змінної;
- кнопки для переходу на інши екрани: «*ТРЕНДЫ*», «АВАРИИ», «*Настройка ПИД-регулятора*» та «*Главный экран*».
- на мнемосхемі також присутні табло з відображенням поточного часу і дати.

Не заглиблюючись в особливості редагування графічних елементів з погляду дизайну, зупинимося на принципах їх конфігурування з погляду мережної взаємодії. Основними параметрами для зв'язку графічного елементу з мережними елементами в протоколі *ModBus* є адреса пристрою у мережі, номер та тип регістра, а також функція, яка застосовується до регістра. Відповідно до структури мережного модуля, який створений в конфігурації ресурсів ПЛК (рис. 4.11), адреси і типи регістрів мають значення, які наведено в табл. 4.3.

Для всіх змінних адреса підпорядкованого *Modbus*-пристрою однакова, це «16». Враховуючи принцип вирівнювання змінних, в структурі пам'яті ПЛК регістрі «0» та «1» задіяні для приймання та відправлення значень окремих бітів (див. табл. 4.3). Інші регістри призначені для зберігання значень параметрів для функціонального блока ПІД-регулювання.

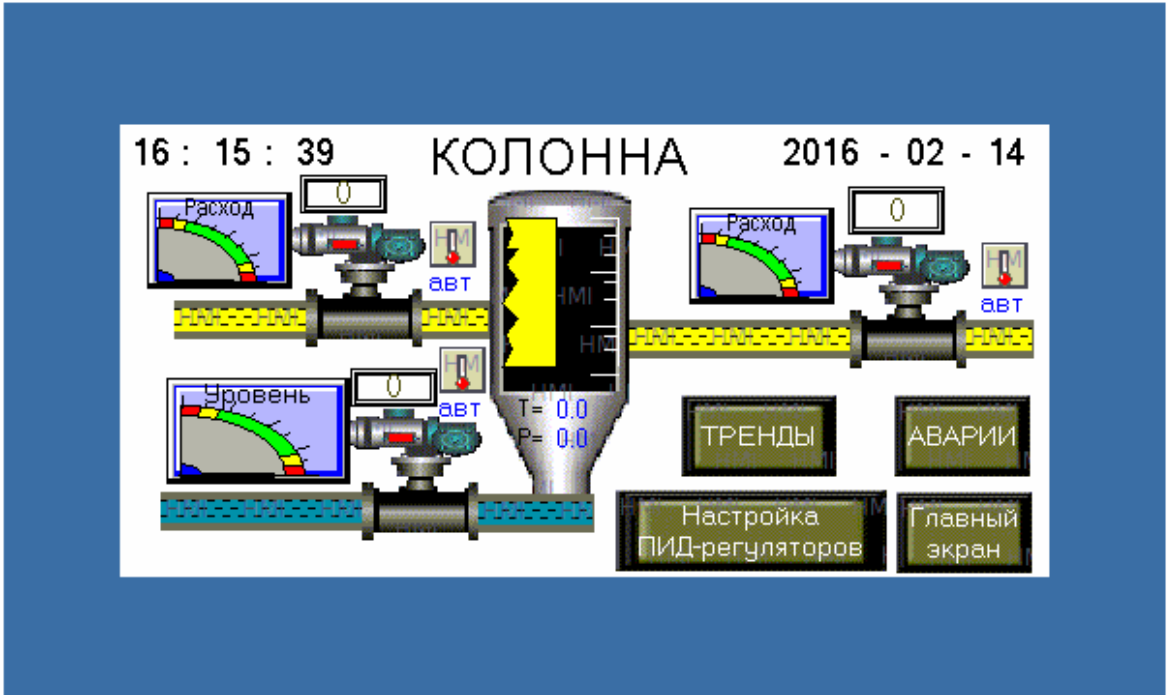


Рисунок 4.12 – Приклад екрана ОП СП-270

Таблиця 4.3 – Характеристики мережних змінних в пам'яті ПЛК

Ім'я змінної в ПЛК	Тип даних в ПЛК	Тип даних в протоколі *ModBus*	Номер елемента		
			регістр	байт	біт
anr_start	BOOL	input_0/8 bits	0	0	0
y_man	BOOL				1
резерв	-	input_1/8 bits	0	1	8–15
plus_work	BOOL	output_1/8 bits	1	2	16
minus_work	BOOL				17
anr_flag	BOOL				18
sv_min	BOOL				19
sv_max	BOOL				20
резерв	-	output_1/8 bits	1	3	24–31
vh_znach_level	DWORD	float	2, 3	5–8	-
ust_level	DWORD	float	4, 5	9–12	-
y_man_set	DWORD	float	6, 7	13–16	-
moshn_upr	DWORD	float	8, 9	17–20	-
zona_nechustv	DWORD	float	10, 11	21–24	-

На рис. 4.13 показаний приклад конфігурування піктограми «Кнопка управления битом», за допомогою якої формується короткий імпульс для включення режиму автоналаштування ПІД-регулятора. На цьому рисунку показано значення адреси пристрою та тип регістра для зберігання значення мережної змінної anr_start. Для кнопки використовується тип «0x», що відповідає у протоколі *Modbus* типу даних «обмотка», у якої адреса дорівнює «0».

На прикладі налаштування піктограми «*Цифровой дисплей*» показано відображення значення мережної змінної vh_znach_level, яка показує значення рівня в колоні (див. рис. 4.14). Тип регістра «4x», що відповідає у протоколі *Modbus* типу даних «регістр-клямка», має адресу «2».

Після закінчення створення всіх елементів екрана збережіть проект. Для перевірки працездатності проекту використайте емулятор панелі для випадку взаємодії з реальним ПЛК. Для цього натисніть на відповідну піктограму в панелі інструментів. Далі, після запуску емулятора, за до-

помогою контекстного меню виберіть потрібний *COM*-порт ПК, до якого підключений ПЛК. Якщо на екрані панелі відображається повідомлення про втрату зв'язку з пристроєм (ПЛК), перевірте номер та параметри налаштування *COM*-порту ПК.

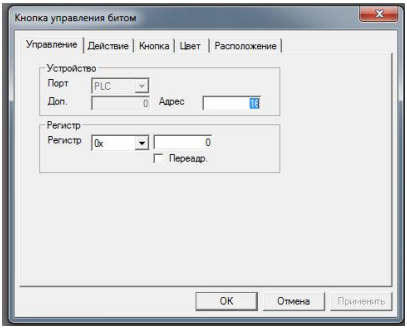


Рисунок 4.13 – Приклад налаштування кнопки на екрані *СП-270*

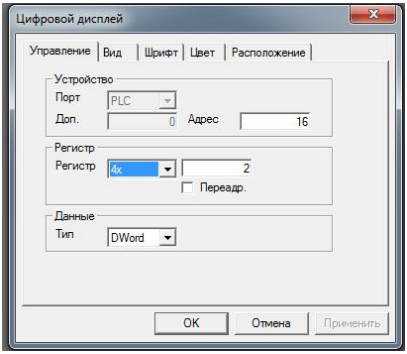


Рисунок 4.14 – Приклад налаштування кнопки на екрані ОП *СП-270*

4.5.3. Налаштування обміну даними між ПЛК110-30 та модулем МСД200

Читання та архівування даних технологічного процесу здійснюється за допомогою модуля архівування даних *МСД-200*, який працює в режимі головного пристрою по відношенню до модуля мережного обміну в ПЛК (див. рис. 4.10). При цьому використовується протокол *ModBus-RTU* по

інтерфейсу *RS-485*. Як і раніше, для налаштування модуля необхідно запустити програму конфігурування модуля. Далі потрібно налаштувати інтерфейс зв'язку та канали для архівування даних технологічних процесів (усього до 64 параметрів). В модулі є можливість підключення чотирьох датчиків з вихідним сигналом 4…20 мА. Основні властивості, технічні характеристики модуля та порядок його конфігурування наведені в [21].

Відмітимо, що функцію архіватора може виконати власне ПЛК. Але для цього потрібно його додатково налагодити – додати до кофігурації його ресурсів програмний модуль Archiever [VAR]. Крім того, можлива передача архівних файлів від ПЛК до ПК за протоколом *ModBus* з використанням інтерфейсу *Ethernet*. Для цього в протоколі передбачена спеціальна функція (№ 20) для передавання файлів по послідовним інтерфейсам та в мережі *Ethernet* [16, 17].

СПИСОК ЛІТЕРАТУРИ

1. Кузьмин И.В., Кедрус В.А. Основы теории информации и кодирования. Уч. Пособие, 2-е изд. перераб. и доп. – К. : «Вища школа», 1986 г. – 238 стр.: ил.

2. Ел. джерело: http://www.iso.org // ГОСТ 34.936-91 (ISO 10039-91) Информационная технология. Локальные вычислительные сети. Определение услуг уровня управления доступом к среде. Дата введения в действие: 01.07.1992.

3. Ел. джерело: http://www.iso.org // RM OSI (Open Systems Interconnection Basic Reference Model) ISO 7498.

4. Ан П. Сопряжение ПК с внешними устройствами: Пер. с англ. – М. : ДМК Пресс, 2001. – 320.: ил.

5. Ел. джерело: http://www.iso.org // Standard RS-232: Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange.

6. Ел. джерело: http://www.iso.org // Standard RS-485: Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems.

7. Олссон Г., Пиани Д. Цифровые системы автоматизации и управления: Пер. с англ. – СПб. : Невский диалект, 2001. – 557 с.: ил.

8. Руководство пользователя по программированию ПЛК в CoDeSys V2.3. – Смоленск : ПК Пролог, 2005. – 453 с.

9. Петров И. В. Программируемые контроллеры. Стандартные языки и приемы программирования / И.В. Петров. – М. : СОЛОН-ПРЕСС, 2008. – 256 с.

10. Методичні вказівки для проведення лабораторних занять з курсу «Програмне забезпечення МПС» / уклад. В.І. Тошинський, І.Г.Лисаченко, І.І. Литвиненко та ін. – Х. : НТУ «ХПІ», 2012. – 56 с.

11. Методичні вказівки для проведення лабораторних занять з курсу «Комп'ютерні мережі» (в двох частинах) для студентів напряму підготовки 050202 «Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання. Частина 1 / Уклад. Подустов М.О., Лисаченко І.Г., Лобойко В.О., Шутинський О. Г. – Х. : НТУ «ХПІ», 2015. – 48 с.

12. Ел. джерело: http://ftp.tiaonline.org // Standard TIA/EIA-602: Data transmission systems and equipment – serial asynchronous automatic dialing and control.

13. Ел. джерело: http://www.siemens.com //TC35i Terminal Siemens Cellular Engine.

14. Ел. джерело: http://www.owen.ua // Контроллер программируемый логический ОВЕН ПЛК150. Паспорт и руководство по эксплуатации – М. : ОВЕН, 98 с.

15. Ел. джерело: http://www.owen.ua // Конфигурирование области ввода/вывода ПЛК. Руководство пользователя – М. : ОВЕН, 119 с.

16. Ел. джерело: http://www.Modbus-IDA.org // MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b.

17. Ел. джерело: http://www.Modbus-IDA.org // MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b.

18. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. – СПб.: Питер, 2012 – 944 с.: ил.

19. Ел. джерело: http://www.owen.ua // Описание протокола обмена между ПЭВМ и приборами ОВЕН. . – М. : ОВЕН, 2012. – 86 с.

20. Ел. джерело: http://www.owen.ua // Модуль ввода аналоговый измерительный МВА8. Руководство по эксплуатации. – М. : ОВЕН, 2012. – 92 с.

21. Ел. джерело: http://www.owen.ua // Модуль дискретного ввода/вывода МДВВ. Паспорт и руководство по эксплуатации. – М. : ОВЕН, 2012. – 47 с.

22. Ел. джерело: http://www.owen.ua // Программа-конфигуратор панели оператора СП200. Руководство пользователя. – М. : ОВЕН, 2012. – 43 с.

23. Ел. джерело: http://www.owen.ua // Преобразователь измерительный регистрирующий. Руководство по эксплуатации, АРАВ. 421451.004 РЭ – М. : ОВЕН, 2012. – 69 с.

Навчальне видання

ЛИСАЧЕНКО Ігор Григорович
ПОДУСТОВ Михайло Олексійович
ЛОБОЙКО В'ячеслав Олексійович
ШУТИНСЬКИЙ Олексій Григорович
БАБІЧЕНКО Анатолій Костянтинович

ПРОМИСЛОВІ МЕРЕЖІ: ТЕОРІЯ І ПРАКТИКА ЗАСТОСУВАННЯ ПРОТОКОЛІВ ТА ІНТЕРФЕЙСІВ

Навчальний посібник
для студентів спеціальності
«Автоматизація та комп'ютерно-інтегровані технології»
денної та заочної форм навчання

Відповідальний за випуск *І.Г. Лисаченко*

Роботу до видання рекомендувала *Н.М. Самойленко*

Редактор *Л.А. Пустовойтова*

План 2016 р., поз. 11

Підп. до друку __.__.2016. Формат 60×84 1/16. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 11. Наклад 300 прим.,
1-ий з-д – 100 прим. Зам. №_____. Ціна договірна.

Видавець та виготовлювач
ТОВ Видавництво «Підручник НТУ „ХПІ"»,
вул. Фрунзе, 21, Харків-2, 61002

Свідоцтво суб'єкта видавничої справи ДК № 3656 від 24.12.2009 р.