МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

В. А. Яланецький

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Комп'ютерний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»

Київ
КПІ ім. Ігоря Сікорського
2020

Відповідальний редактор: Петро КРАВЕЦЬ, к.т.н., доц. каф. АУТС

Рецензент: Артем ВОЛОКИТА, к.т.н., доц. каф. ОТ

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 3 від 05.11.2020 р.)
за поданням Вченої ради факультету Інформатики та обчислювальної техніки
(протокол № 10 від 18.05.2020 р.)

Електронне мережне навчальне видання

Валерій ЯЛАНЕЦЬКИЙ, ст. викладач

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Комп'ютерний практикум

Проектування інформаційних систем: комп'ютерний практикум [Електронний ресурс]: навч. посіб. для студ. освітньої програми «Інтегровані інформаційні системи» спеціальності 126 «Інформаційні системи та технології» / КПІ ім. Ігоря Сікорського; уклад.: В.А. Яланецький. – Електронні текстові дані (1 файл: 3.1 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 138с.

Навчальний посібник призначений для студентів освітньої програми «Інтегровані інформаційні системи» спеціальності 126 «Інформаційні системи та технології» кафедри автоматики та управління в технічних системах всіх форм навчання. Тематика посібника відповідає робочій програмі з дисципліни «Проектування інформаційних систем», яка є обов’язковою дисципліною у навчальному плані підготовки бакалаврів зі спеціальності 126. В навчальному посібнику викладені базові теоретичні відомості та лабораторні роботи до дисципліни «Проектування інформаційних систем». Лабораторний практикум є можливість виконувати як у фізичних лабораторіях кафедри так і дистанційно на власних робочих станціях. Лабораторні роботи виконуються у програмному комплексі CoDeSys. В посібнику наводяться рекомендації з використання цього програмного комплексу для задач проектування інформаційних систем а саме: програмне моделювання об′єктів керування, застосування зворотніх зв′язків, програмування комунікаційного протоколу MODBUS, розробка програм для HMI панелей та АРМО. Посібник може бути корисний студентам відповідних спеціальностей при вивченні дисциплін, пов’язаних із технологіями моделювання та проектування інформаційних систем керування.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ ТА ТЕРМІНІВ 6

ВСТУП 7

1 ЛАБОРАТОРНА РОБОТА № 1 10

1.1 Постановка задачі розробки програми АСК 11

1.2 Теоретичні відомості про мову SFC 11

1.3 Приклад виконання роботи 17

1.4 Контрольні запитання 23

1.5 Контрольне завдання 25

2 ЛАБОРАТОРНА РОБОТА № 2 26

2.1 Постановка (скорегована) задачі розробки програми АСК 27

2.2 Теоретичні відомості 27

2.3 Приклад виконання роботи 28

2.4 Контрольні запитання 31

2.5 Контрольне завдання 33

3 ЛАБОРАТОРНА РОБОТА № 3 34

3.1 Постановка задачі розробки програмної моделі об'єкту 34

3.2 Теоретичні відомості про синтаксис мови VBScripts 36

3.3 Приклад виконання роботи 39

3.4 Контрольні запитання 44

3.5 Контрольне завдання 44

4 ЛАБОРАТОРНА РОБОТА № 4 45

4.1 Постановка задачі застосування протоколу MODBUS 45

4.2 Теоретичні відомості про стандарт MODBUS 47

4.3 Приклад виконання роботи 49

4.4 Алгоритм ручної імітації поведінки ОК 56

4.5 Контрольні запитання 58

4.6 Контрольне завдання 58

5 ЛАБОРАТОРНА РОБОТА № 5 60

5.1 Постановка задачі розробки людино-машинного інтерфейсу..... 60
5.2 Теоретичні відомості про HMI..... 61
5.3 Приклад виконання роботи 62
5.4 Контрольні запитання 72
5.5 Контрольне завдання..... 72
6 ЛАБОРАТОРНА РОБОТА № 6..... 73
6.1 Постановка задачі розробки об'єктної програми керування..... 73
6.2 Теоретичні відомості..... 73
6.3 Приклад виконання роботи 73
6.4 Контрольні запитання 77
6.5 Контрольне завдання..... 77
7 ЛАБОРАТОРНА РОБОТА № 7..... 78
7.1 Постановка задачі проектування регулятора 78
7.2 Теоретичні відомості про регулятори 78
7.3 Приклад виконання роботи 79
7.4 Контрольні питання 84
7.5 Контрольне завдання..... 84
8 ЛАБОРАТОРНА РОБОТА № 8..... 85
8.1 Постановка задачі розробки WEB-панелі..... 85
8.2 Теоретичні відомості..... 86
8.3 Приклад виконання роботи 86
8.4 Приклад роботи WEB-візуалізації у браузері 91
8.5 Контрольні запитання 94
8.6 Контрольне завдання..... 94
9 ЛАБОРАТОРНА РОБОТА № 9..... 95
9.1 Постановка задачі розробки АРМО 95
9.2 Базові теоретичні відомості про Trace Mode..... 98
9.3 Приклад виконання роботи 98
9.4 Приклад розробки АРМО в Trace Mode..... 103

9.5 Контрольні запитання 113
9.6 Контрольне завдання 113
10 ЛАБОРАТОРНА РОБОТА № 10 114
10.1 Постановка задачі взаємодії АРМО із базами даних 114
10.2 Теоретичні відомості про БД 114
10.3 Приклад виконання роботи 116
10.4 Робота з SQL-запитами 122
10.5 Створення запитів на запис в БД 128
10.6 Інтеграція з СУБД 129
10.7 Виведення інформації з БД до ГЕ «База данных» 134
10.8 Контрольні запитання 137
10.9 Контрольне завдання 137
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 138

СПИСОК СКОРОЧЕНЬ ТА ТЕРМІНІВ

Скорочення:

ЛР – лабораторна робота (комп'ютерного практикуму)

ПЛК – мікропроцесорний програмований логічний контролер

ВК – віртуальний контролер (наприклад, CoDeSys Control Win)

ПК – персональний комп'ютер

ВП – виконавча цільова платформа (ВК, ПК, ПЛК, станція, сервер, тощо)

МЕК – міжнародна електротехнічна організація зі стандартизації

ПЗ – програмне забезпечення (програмні засоби)

ООП – парадигма об'єктно-орієнтованого програмування

HMI – людино-машинний інтерфейс (операторська панель, станція)

ФБ – функціональний блок в мовах програмування стандарту МЕК

ЛКМ, ПКМ – ліва та права кнопки маніпулятора «Миша»

ОК – об'єкт керування; ОА – об'єкт автоматизації

САР, САК – система автоматичного регулювання (керування)

ОСЛК – однотактна система логічного керування

АСК – автоматизована система керування

MS та SL – MASTER та SLAVE пристрої промислових мереж

Терміни:

SoftLogic – програмна технологія програмування ПЛК

Gateway-сервер – сервер емуляції ВП та ВК

Симуляція – режим роботи програмної моделі в реальному часі

ВСТУП

В навчальному посібнику викладені цілі, завдання, теоретичні відомості, приклади виконання до курсу лабораторних робіт (ЛР) з дисципліни «Проектування інфоормаційних систем» (ПрІС), а також порядок їх захисту на кафедрі «Автоматики та управління в технічних системах» факультету «Інформатики та обчислювальної техніки» Національного технічного університету України «КПІ ім. Ігоря Сікорського» для студентів спеціальності 126 «Інформаційні системи та технології».

Навчальний посібник призначений орієнтований на двосеместровий курс дисципліни (два кредитних модулі) для всіх форм навчання. Кожен кредитний модуль містить 5 лабораторних робіт. Для денної форми навчання обов'язковим є виконання всіх п'яти лабораторних робіт. Для заочної форми – перших трьох лабораторних робіт.

Комплекс ЛР з дисципліни ПрІС є продовженням вивчення та практичного засвоєння сучасних інформаційних технологій, засобів та програмних інструментів щодо моделювання та програмування автоматизованих систем керування (АСК) в межах навчального процесу. Дисципліна ПрІС складається з двох кредитних модулів: ПрІС-1 та ПрІС-2 відповідно.

Мета лабораторних робіт:

- заглиблення у вивченні середовища програмування CoDeSys;
- ознайомлення із розширеними принципами програмування ПЛК технологічними мовами згідно стандарту МЕК 61131-3 (МЕК);
- закріплення теоретичних навичок з інформаційних систем;
- закріплення теоретичних навичок з моделювання та програмування локальних систем керування;

Завдання лабораторних робіт:

- засвоїти методику розробки програм мовами стандарту МЕК;
- вивчити принцип складання програм в CoDeSys;

- практична реалізація контурів керування в програмі для ПЛК.

Протягом виконання ЛР студент повинен отримати навички:

1) роботи з програмним забезпеченням CoDeSys;

2) проектування керуючих програм для ВК в CoDeSys;

3) використання мов стандарту МЕК;

4) складання програмних моделей технологічних процесів;

5) складання програмних моделей замкнених систем керування.

Суть кожної ЛР полягає в розв'язанні типової задачі програмування контуру керування (або регулювання певного параметру) в межах САР.

Протягом виконання кожної ЛР студент самостійно повинен:

- розробити алгоритм розв'язанні задачі;
- створити керуючу програму засобами CoDeSys технологічними мовами зі стандарту МЕК;
- скомпілювати та завантажити розроблену програму до цільової ВП (ВК або ПЛК) й виконати її запуск;
- перевірити правильність роботи програми на ВП;
- оформити звіт та захистити ЛР.

Кожна ЛР виконується і оформлюється відповідно до завдання на ЛР, варіант якої можна знайти в таблиці або ж завдання видається викладачем студентові особисто в електронному (друкованому) вигляді.

Комплекс ЛР ПрІС-1 складається із наступних робіт:

1) Програмування АСК мовою послідовних операцій SFC;

2) Програмування АСК із додаванням зворотних зв′язків;

3) Створення скрипту імітатора об'єкта керування;

4) Застосування протоколу MODBUS для комунікації ВК з ОК;

5) Розробка локального людино-машинного інтерфейсу HMI.

Комплекс ЛР ПрІС-2 складається із наступних робіт:

1) Об'єктно-орієнтована програма керування;

2) Проектування регулятора та програмного задавача;

3) WEB-диспетчеризація;

4) Розробка АРМО в Trace Mode та його зв'язок із ВК;

5) Взаємодія АРМО із зовнішніми базами даних.

Навчальний посібний містить велику кількість прикладів виконання робіт що надає студентам та інших слухачам протягом вивчення дисципліни ПрІС самостійно, якісно та швидко виконувати індивідуальні завдання.

Навчальний посібний є поглибленим навчальним документом для подальшого вивченні інформаційних технологій в сфері моделювання складних, розподілених та інтелектуальних систем керування, синтезу регуляторів, автоматизованих систем, а також проектування інформаційно-керуючих систем керування.

1 ЛАБОРАТОРНА РОБОТА № 1

Тема: Програмування АСК мовою послідовних операцій SFC

Мета: Знайомство з основами програмного проектування ВК мовою послідовних інструкцій SFC без зворотних зв′язків.

Завдання:

1) Засвоїти методику розробки програми мовою SFC для ВК на прикладі АСК для ОА (без зворотних зв′язків), який зображений на рис. 1.1.

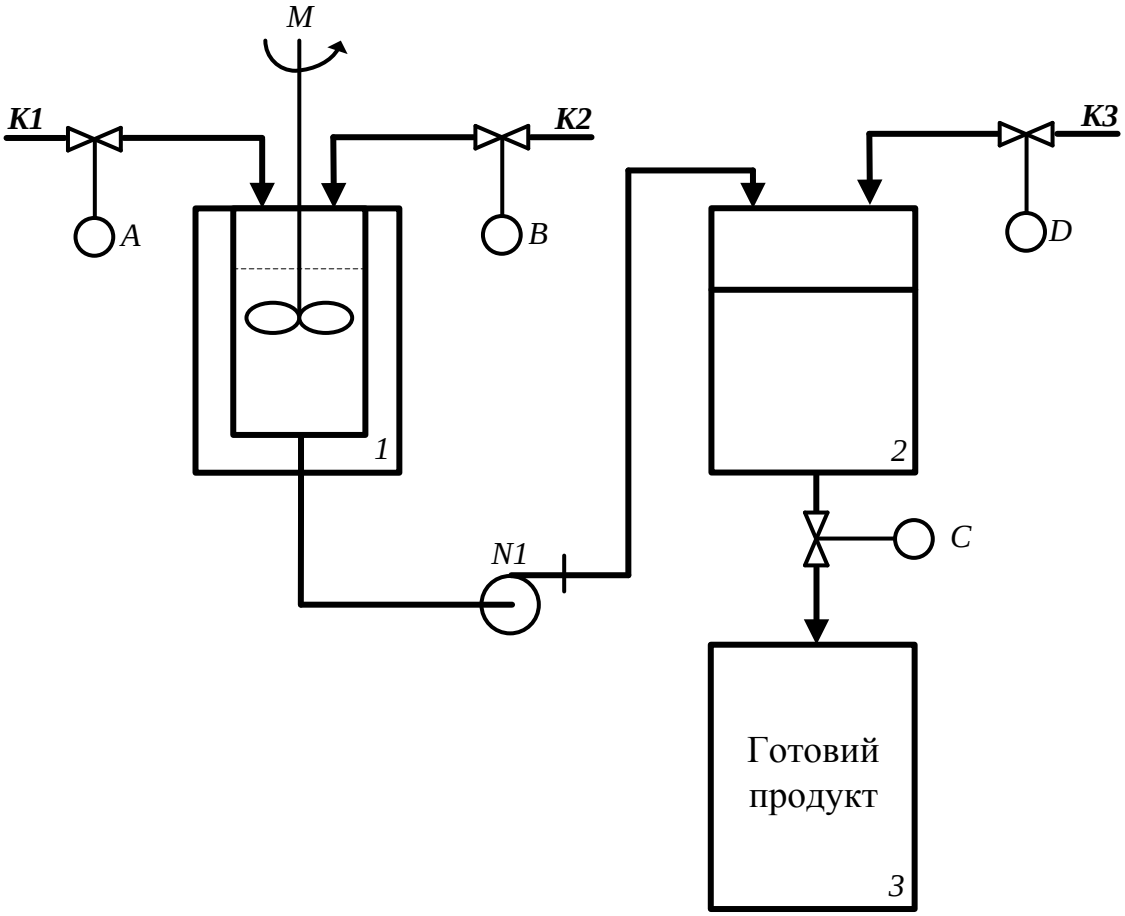


Рис. 1.1. Приклад АСК типового ОА

2) Виконати контрольне завдання до ЛР №1.

1.1 Постановка задачі розробки програми АСК

Розробити АСК із наступними вимогами:

1) Оголосити словник змінних моделі ОА та окремо змінних АСК;

2) Розробити програмну модель ОА мовою ST;

3) Розробити керуючу програму мовами SFC/ST із додаванням у єдиний проект програмної моделі ОА без зворотних зв′язків (без давачів).

Структура віртуального лабораторного стенду №1

Лабораторний стенд на базі ПК складається із наступних елементів:

1) Середовище програмування CoDeSys;

2) Модель об'єкту автоматизації (ОА) – імплементована розробником у код CoDeSys спрощена програмна модель ОА;

3) ВК – **CoDeSys Control Win**.

Структура лабораторного стенду наведена на рис. 1.2.

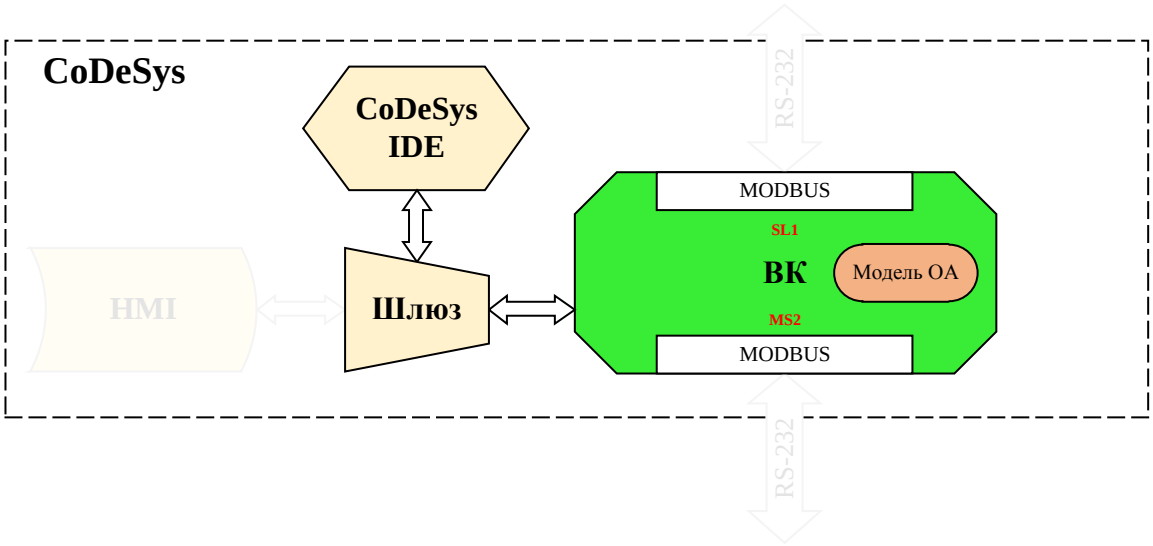


Рис. 1.2. Структура стенду ЛР №1

1.2 Теоретичні відомості про мову SFC

Мова SFC (Sequence Function Chart) – це графічна мова, яка використовується для опису послідовних операцій. Процес представляється у вигляді набору певних кроків, зв'язаних переходами. До кожного переходу

прикріплена логічна умова. Дії всередині кроків описані більш детально за допомогою інших мов (зазвичай – ST).

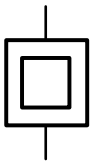
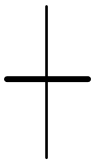
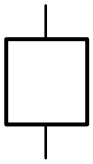
SFC програма – це графічний набір кроків і переходів, з'єднаних разом спрямованими зв'язками. Для позначення сходжень і розходжень використовуються множинні зв'язки. Деякі частини програми можуть бути відокремлені та представлені в основній схемі одним символом – макрокроком. Ось основні графічні правила для SFC:

1) Кроки не можуть йти підряд.

2) Переходи не можуть йти підряд.

Основні компоненти SFC представлені в таблиці 1.1.

Таблиця 1.1 – Основні компоненти SFC

Компонент	**Опис**	**Компонент**	**Опис**
	Початковий крок		Перехід
	Крок		Стрибок на крок

Крок представляється одиночним квадратом. Кожному кроку присвоюється номер, який написаний всередині квадрата. Основний опис кроку записується всередині прямокутника, приєднаного до символу кроку. Це вільний коментар, який не є частиною мови. Вищенаведена інформація називається «**Рівнем 1 кроку**» (рис. 1.3).

Під час роботи активний крок позначається маркером (виділяється).

Рис. 1.3. Приклад Рівня 1 кроку

Початкова ситуація програми SFC описується початковими кроками. Початковий крок позначається графічно символом з подвійною рамкою. Після запуску програми маркер автоматично встановлюється на кожний початковий крок (рис. 1.4).

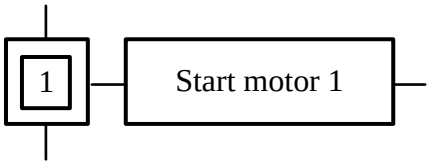


Рис. 1.4. Початковий крок SFC програми

У кожного кроку є атрибути. Вони можуть бути використані в будь-якій іншій мові в будь-якому місці програми:

GSnnn.x – активність кроку (логічна змінна);

GSnnn.t – тривалість активного стану кроку (таймер), де **nnn** – крок.

Переходи представлені горизонтальними смужками, які перетинають лінії зв'язку. Кожному переходу присвоєний номер, який йде після символу переходу. Опис переходу (коментар) розташовується праворуч від символу переходу. Вищенаведена інформація – це «**Рівень 1 переходу**» (рис. 1.5).

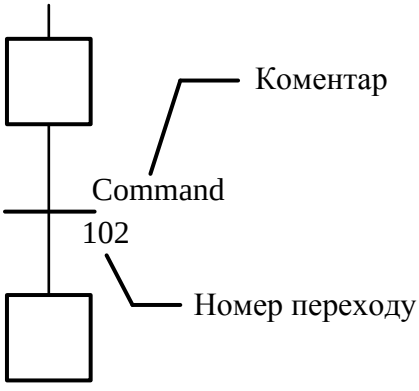


Рис. 1.5. Рівень 1 переходу

Для зв'язку кроків та переходів використовуються одиночні лінії. Це орієнтовані зв'язки. Коли орієнтація не задана явно, зв'язок орієнтований зверху вниз (рис. 1.6).

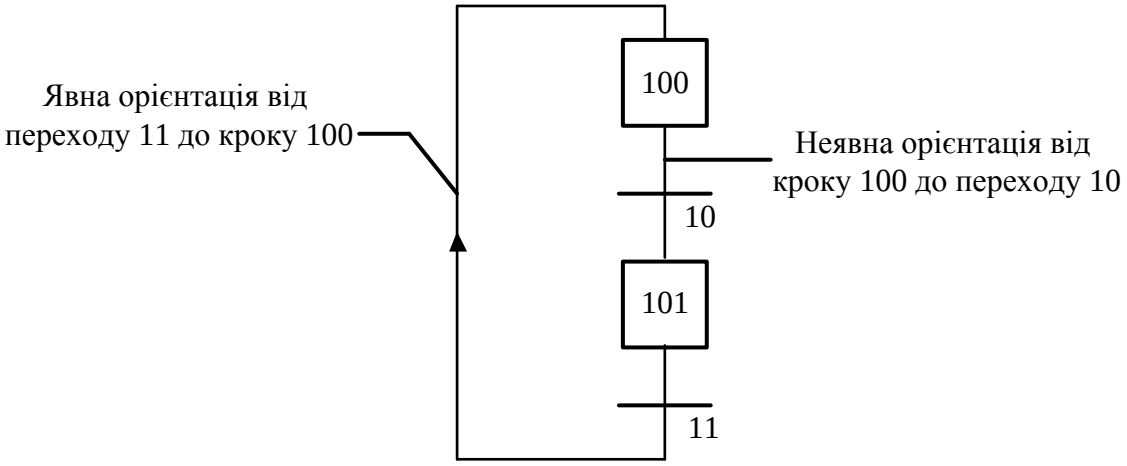


Рис. 1.6. Орієнтовані зв'язки

Символ стрибка може бути використаний, щоб визначити лінію зв'язку від переходу до кроку, не зображуючи лінію. Символ стрибка повинен мати номер кроку призначення (рис. 1.7).

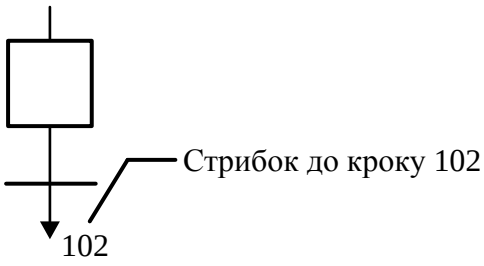


Рис. 1.7. Символ стрибка

Зв'язок від кроку до переходу не можна представити за допомогою символу стрибка.

Розходження – це множинні зв'язки від одного символу SFC (кроку або переходу) до інших. Сходження – це множинні зв'язки від більше, ніж одного символу SFC до одного іншого символу. Сходження та розходження можуть бути одиночними або подвійними.

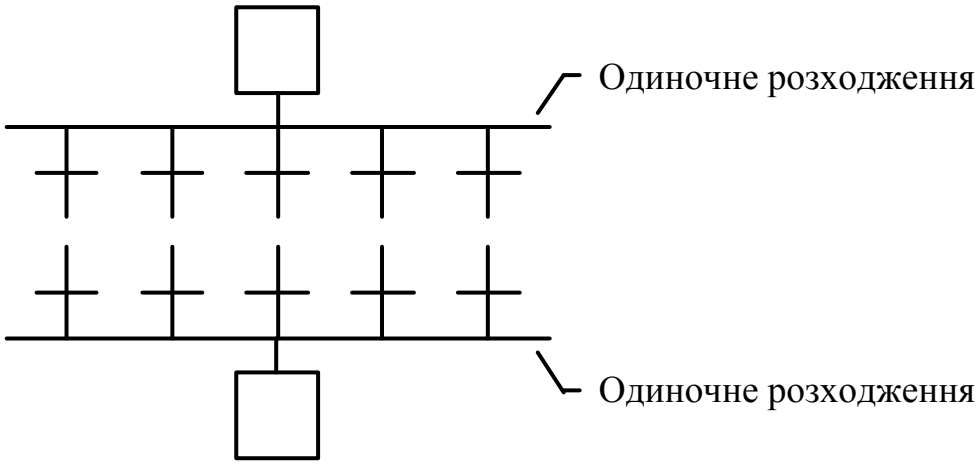


Рис. 1.8. Одиночне розходження

Одиночне розходження – це множинний зв'язок від одного кроку до декількох переходів. Воно дозволяє маркеру активно пройти по одній з безлічі гілок. Одиночне сходження – це множинний зв'язок від декількох переходів до одного і того самого кроку, що зазвичай використовується для того, щоб об'єднати декілька гілок SFC, які розпочалися з одиночного розходження. Одиночні розходження і сходження позначаються одиночними горизонтальними лініями (рис. 1.8).

ЗАУВАЖЕННЯ. Умови, які приєднані до переходів, не є взаємно виключними. Для того, щоб програма пішла по одній гілці, потрібно явно визначити виключність умов переходу.

Подвійне розходження – це множинний зв'язок від одного переходу до декількох кроків. Він відповідає паралельній роботі процесу. Подвійне сходження – це множинний зв'язок від декількох кроків до одного і того самого переходу. Подвійне сходження, зазвичай, використовується для того, щоб об'єднати декілька гілок SFC, які розпочалися з подвійного розходження. Подвійні розходження та сходження позначаються подвійними горизонтальними лініями (рис. 1.9).

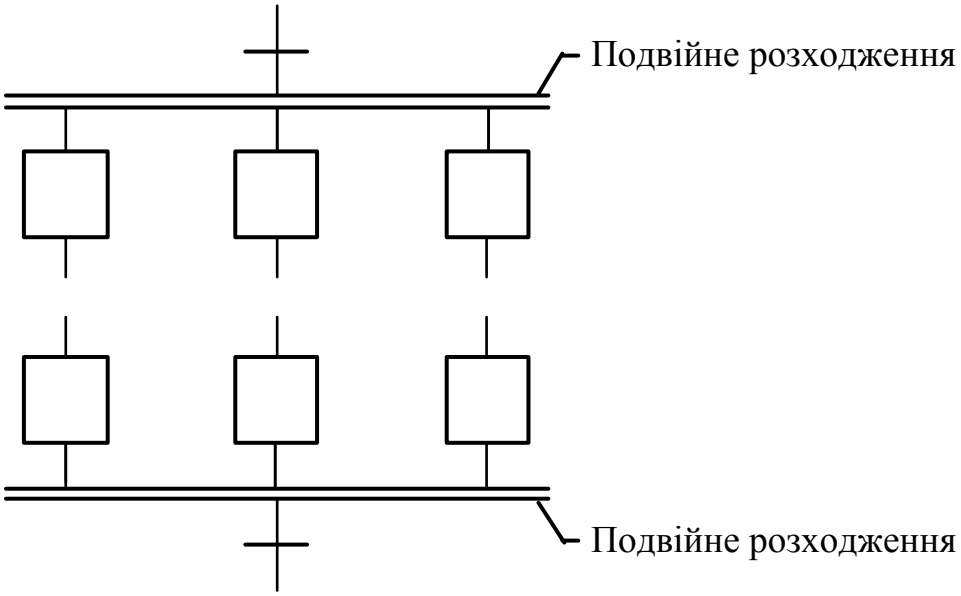


Рис. 1.9. Подвійне розходження

Рівень 2 кроку SFC являє собою детальний опис дій в період активності кроку, де можна використовувати текстові доповнення мови SFC або мову ST. Основні типи дій: булеві дії; імпульсні дії, описані на ST (дії типу «P»); дії, які не зберігаються, описані на ST (дії типу «N»); SFC дії.

Булеві дії присвоюють значення логічної змінної при активізації кроку. Логічні змінні можуть бути вихідними або внутрішніми. Їм присвоюється значення кожний раз, коли крок становиться активним або перестає бути активним. Синтаксис основних логічних дій наведений в табл. 1.2.

Таблиця 1.2 – Синтаксис основних логічних дій

Логічна дія	Опис
`<boolean_variable> (N); <boolean_variable>;`	Присвоїти змінній сигнал активності кроку
`/ <boolean_variable>;`	Присвоїти змінній заперечення сигналу активності кроку
`<boolean_variable> (S);`	Присвоїти змінній значення TRUE, коли крок стає активним
`<boolean_variable> (R);`	Присвоїти змінній значення FALSE, коли крок стає активним

1.3 Приклад виконання роботи

Розробити програму на мові SFC, яка реалізує алгоритм приведеної нижче послідовності дій АСК:

1. При натисненні кнопки «**СТАРТ**» розпочати процес, при цьому відкрити клапан **A** і завантажувати компонент ***K1*** протягом 10 с в ємність *1*.
2. По закінченні завантаження компонента ***K1***, при працюючій мішалці **M** розпочати завантаження компонента ***K2***, відкривши клапан **B**.
3. Завантаження здійснювати протягом 15 с.
4. По закінченні завантаження компонента ***K2*** суміш продовжувати перемішувати **ще протягом 10 с**.
5. Потім суміш перекачати в ємність 2, включивши насос **N1**. Спорожнення ємності *1* триває 25 с.
6. В ємність *2* завантажувати компонент ***K3*** протягом 10 с, відкривши клапан **D**.
7. Суміш повинна витримуватись протягом 15 с, після чого відкрити клапан **C** та вивантажити готовий продукт з ємності *2* в ємність *3*. Спорожнення ємності *2* триває 35 с.
8. Підготувати лінію для приготування нової партії продукту.

1) Оголосити словник змінних та параметрів ємностей ОА та ТЗА із зазначенням номінальних значень (табл. 1.3-1.5).

2) Створити проект **«lab_sfc»** аналогічно крокам 1-6 ЛР №1 МПС-1.

3) Реалізувати програмну модель ОА (зміни стану, спрацювання давачів або процес накопичення рідини в ємностях).

Таблиця 1.3 – Змінні ВОА

Змінна	Пояснення
q1	поточний об'єм компоненту у ємності 1 [л]
q2	поточний об'єм компоненту у ємності 2 [л]
h1	поточний рівень компоненту у ємності 1 [дм]
h2	поточний рівень компоненту у ємності 2 [дм]
overflow1	контроль ємності 1 – переповнена, 0 – ні [1/0]
overflow2	контроль ємності 2 – переповнена, 0 – ні [1/0]
valve_in1	сигнал закр./відкрит. впускного клапану **A** [1/0]
valve_in2	сигнал закр./відкрит. впускного клапану **B** [1/0]
valve_in3	сигнал закр./відкрит. впускного клапану **D** [1/0]
valve_out1	сигнал закр./відкрит. випускного клапану **C** [1/0]
mixer	сигнал вмикання/вимикання мішалки **M** [1/0]
pump	сигнал вмикання/вимикання насосу **N1** [1/0]
h1_max	макс. рівень компоненту у ємності 1 [дм]
h2_max	макс. рівень компоненту у ємності 2 [дм]

Таблиця 1.4 – Параметри ВОА

Параметр	Ном.	Пояснення
r_tank1	10	радіус циліндричної ємності 1 [дм]
r_tank2	15	радіус циліндричної ємності 2 [дм]
q_max1	10000	максимальний об'єм ємності 1 [л]
q_max2	15000	максимальний об'єм ємності 2 [л]
r_hole1	0.5	радіус сточного отвору ємності 1 [дм]
r_hole2	0.5	радіус сточного отвору ємності 2 [дм]
q_in1	100	макс. швидкість притоку через клапан **A** [л/сек.]
q_in2	100	макс. швидкість притоку через клапан **B** [л/сек.]
q_in3	100	макс. швидкість притоку через клапан **D** [л/сек.]
q_out1		поточна швидкість стоку через отвір ємн. 1 [л/сек.]
q_out2		поточна швидкість стоку через клапан **C** [л/сек.]
q_in4		поточна швидкість притоку через насос **N1** [л/сек.]

Допоміжні параметри ОА сґруповано у табл. 1.5.

Таблиця 1.5 – Допоміжні параметри ВОА

Параметр	Ном.	Пояснення
dt	0.1	тривалість одного такту «життя» ОА [сек.]
g	98	прискорнення вільного падіння [дм/сек.^{2}]
pi	3.1415	константа π

Програмний імітатор ОА має бути імплементований на мові програмування ST та розміщений у окремому ФБ (це дозволить у наступній ЛР зручно видалити цей імітатор з коду, залишивши в програмі CoDeSys лише алгоритм керування АСК).

Для цього в проекті створюємо новий ФБ, який назвемо, наприклад, **Model_OA** (рис. 1.10). В розділі декларування **Model_OA** помістити наступний код:

```
FUNCTION_BLOCK Model_OA
VAR_INPUT
     valve_in1, valve_in2, valve_in3, valve_out1: BOOL;
     mixer, pump: BOOL;
END_VAR
VAR_OUTPUT
END_VAR
VAR
     h1, h2, q1, q2: REAL: = 0.0;
     r_tank1, r_tank2: REAL: = 10.0;
     r_hole1, r_hole2: REAL: = 0.5;
     in1, in2, in3: REAL: = 0.0;
     q_in4, q_out1, q_out2: REAL: = 0.0;
END_VAR
VAR CONSTANT
     q_in1: REAL: = 2.0;
     q_in2: REAL: = 4.0;
     q_in3: REAL: = 6.0;
     pi: REAL: = 3.1415;
     g: REAL: = 98.0;
END_VAR
```

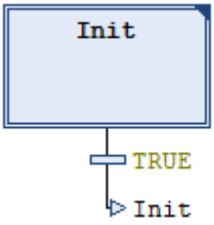


Рис. 1.10. Реалізація ФБ Model_OA

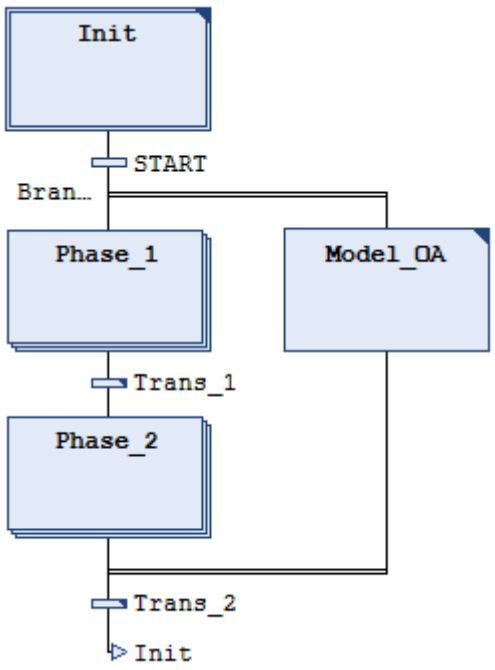
У ФБ **Init** помістити наступний код мовою ST (модель ОА):

```
in1: = q_in1 * BOOL_TO_INT(valve_in1);
in2: = q_in2 * BOOL_TO_INT(valve_in2);
in3: = q_in3 * BOOL_TO_INT(valve_in3);
h1 : = q1/(pi * r_tank1 * r_tank1);
h2 : = q2/(pi * r_tank2 * r_tank2);
IF q1 <= 0 THEN q_out1 := 0;
ELSE
     q_out1 := SQRT(2 * g * h1) * pi * r_hole1 * r_hole1 *
BOOL_TO_INT(pump);
     q_in4 := q_out1;
END_IF
IF q2 <= 0 THEN q_out2 := 0;
ELSE
     q_out2 := SQRT(2 * g * h2) * pi * r_hole2 * r_hole2 *
BOOL_TO_INT(valve_out1);
END_IF
q1 := q1 + (in1 + in2 - q_out1);
q2 := q2 + (in3 + q_in4 - q_out2);
IF q1 <= 0 THEN q1 := 0; h1 : = 0;
END_IF
IF q2 <= 0 THEN q2 : = 0; h2 : = 0;
END_IF
```

4) Створити ФБ **lab1_fb** на мові SFC. В розділі декларування **lab1_fb** помістити наступний код:

```
FUNCTION_BLOCK lab1_fb
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
     START: BOOL: = FALSE;
     v_in1, v_in2, v_in3, v_out1: BOOL: = FALSE;
     m, n1: BOOL: = FALSE;
     model_fb: Model_OA; // ФБ моделі ОА
END_VAR
```

5) В розділі реалізації потрібно набрати необхідну програму (схему SFC) керування (рис. 1.11-1.13).



Елемент	*Код мовою ST*
Init	START:=TRUE; v_in1:=FALSE; v_in2:=FALSE; v_in3:=FALSE; v_out1:=FALSE; m:=FALSE; n1:=FALSE;
START	START
Phase_1	див. рис. 1.12
Trans_1	Step3.t>=T#10S;
Phase_2	див. рис. 1.13
Trans_2	Step7.t>=T#15S;
Model_OA	model_fb(valve_in1:=v_in1, valve_in2:=v_in2, valve_in3:=v_in3, valve_out1:=v_out1, mixer:=m, pump:=n1);

Рис. 1.11. Схема програми керування АСК мовами SFC та ST

За замовчуванням, створюється порожня діаграма, яка містить початковий крок **«Init»** та відповідний перехід **«Trans0»** до **«Init»**.

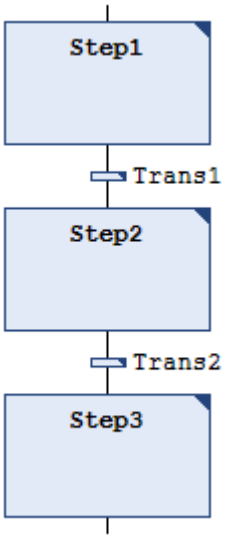
За допомогою команди вставки кроку та переходу **«Step-Transition (after)»**, команди вставки паралельної гілки «**Insert branch right**», команди вставки макро «**Insert macro (after)**» реалізуємо схеми (рис. 1.11-1.13).

Для програмування Кроку необхідно двічі натиснути на зображення відповідного кроку і в новому вікні вибрати мову реалізації **ST**. У відповідності зі схемами на рис. 1.11-1.13 набрати код для кожного з кроків.

Для програмування Умови переходу можна задати або Змінну з проекту (натискаємо багатокрапку та обираємо, наприклад START) або кодом, наприклад мовою ST (двічі натискаємо ЛКМ на горизонтальну лінію зліва від назви переходу).

Схема АСК (рис. 1.11) містить блоки **Phase_1** та **Phase _2**. Це макроелементи, які містять реалізацію певної стадії алгоритму керування АСК.

Для того, щоб відкрити внутрішню реалізацію макро-елемента необхідно двічі натиснути ЛКМ на необхідному блоці. Щоб повернутись на вищий рівень схеми необхідно натиснути кнопку «**Zoom out of macro**» , яка знаходиться на верхній панелі інструментів CoDeSys.

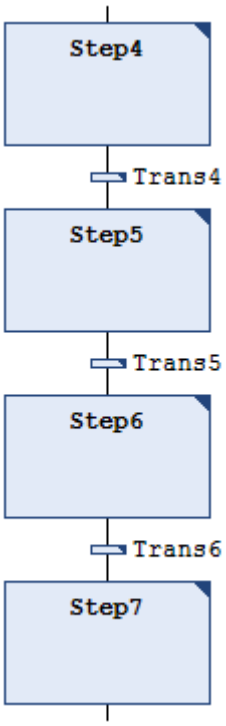


Елемент	*Код мовою ST*
Step1	`v_in1:=TRUE;`

Step2	`v_in1:=FALSE;`
	`v_in2:=TRUE;`
	`m:=TRUE;`

Step3	`v_in2:=FALSE;`
`Trans_1`	`Step1.t>=T#10S;`
`Trans_2`	`Step2.t>=T#10S;`

Рис. 1.12. Реалізація макро-елемента Phase_1



Елемент	*Код мовою ST*
Step4	`m:=FALSE;` `n1:=TRUE;`
Step5	`n1:=FALSE;` `v_in3:=TRUE;`
Step6	`v_in3:=FALSE;`
Step7	`v_out1:=TRUE;`
`Trans_4`	`Step4.t>=T#10S;`
`Trans_5`	`Step5.t>=T#10S;`
`Trans_6`	`Step6.t>=T#15S;`

Рис. 1.13. Реалізація макро-елемента Phase_2

6) Додати в тіло головної програми PLC_PRG виклик ФБ **lab1_fb**:

Декларація:

```
PROGRAM PLC_PRG
VAR
     pba_fb: lab1_fb;
END_VAR
```

Реалізація:

```
pba_fb();
```

Запуск проекту на виконання

Запускається проект за допомогою «**Online → Login**» і можна спостерігати за значеннями змінних програми. Обов'язково перед цим перевірити, чи працюють **Gateway**-сервер та ВК (ЛР №1 МПС-1).

Результати роботи АСК наведені на рис. 1.14.

1.4 Контрольні запитання

1) Призначення мови SFC. Що являє собою SFC програма?

2) За допомогою чого описуються дії всередині кроків?

3) Що використовується для зображення сходжень та розходжень?

4) Назвати основні компоненти мови SFC.

5) Яким графічним елементом зображується крок в SFC програмі?

6) Що використовується для зображення переходів?

7) Дати визначення «розходження». Які існують види розходжень?

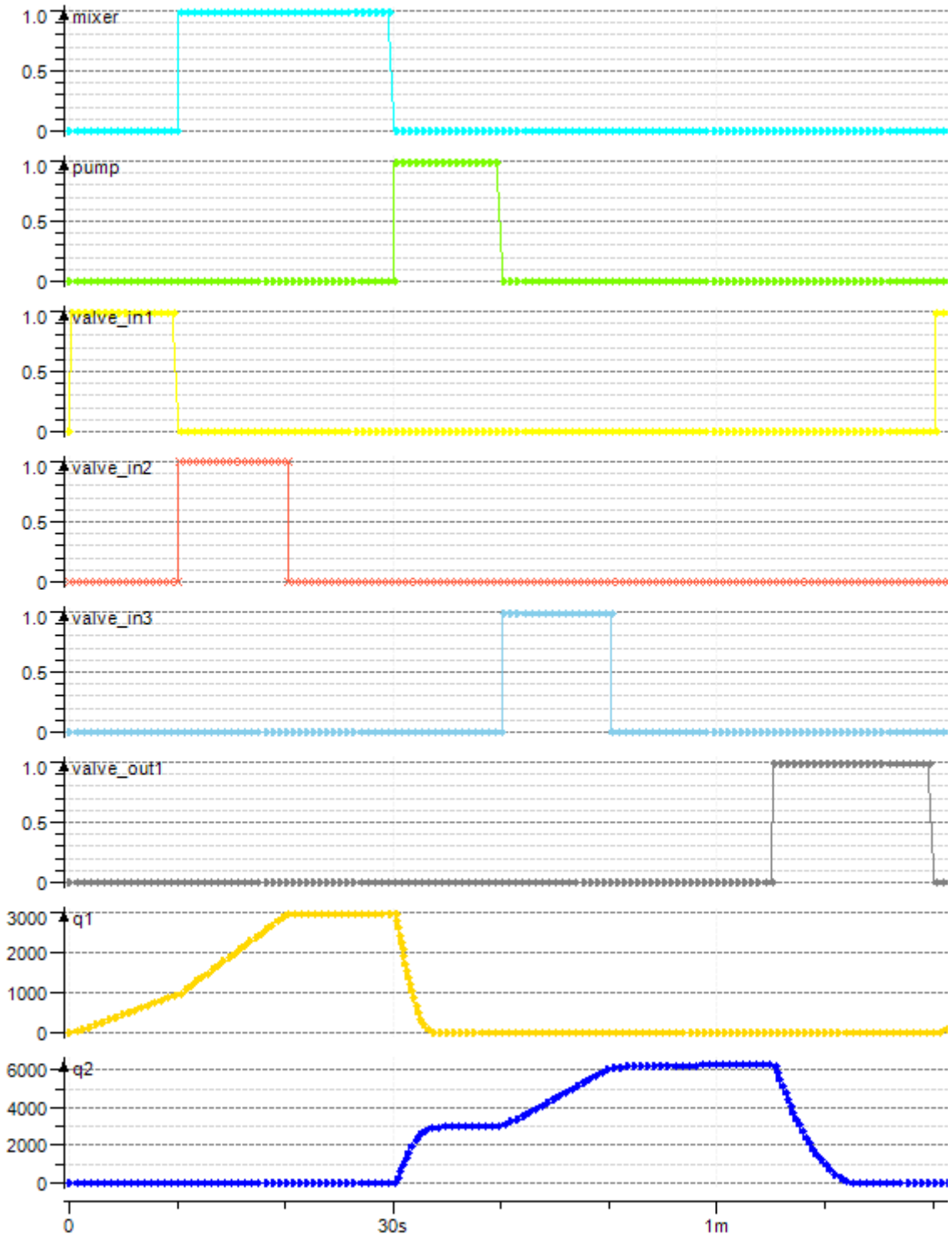


Рис. 1.14. Часова діаграма роботи АСК.

1.5 Контрольне завдання

1) Додати до вихідного проекту новий об'єкт ***Додаток*** з назвою відповідно до шаблону **PIP_LR01**. Встановити його активним.

2) За власним варіантом у файлі «**ПрІС. ЛР. БІЗ.pdf**» віднайти та виконати індивідуальне завдання до ЛР №1:

- записати алгоритм функціонування АСК та самостійно сформувати словники змінних й параметрів ОА та АСК.
- розробити програму керування технологічною схемою.
- імплементувати в програму **імітацію задавальних уставок та спрощену модель ОА (імітація зміни значень давачів)**.

Результати ЛР подати у вигляді графіків вхідних та вихідних сигналів, побудованих в **CoDeSys.Trace**.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

2 ЛАБОРАТОРНА РОБОТА № 2

Тема: Програмування АСК із додаванням зворотних зв′язків

Мета: Знайомство з основами програмного проектування АСК мовою SFC зі зворотними зв′язками (наявність давачів).

Завдання:

1) Засвоїти методику розробки програми мовою SFC для ВК на прикладі АСК для ОА (зі зворотними зв′язками), який зображений на рис. 2.1.

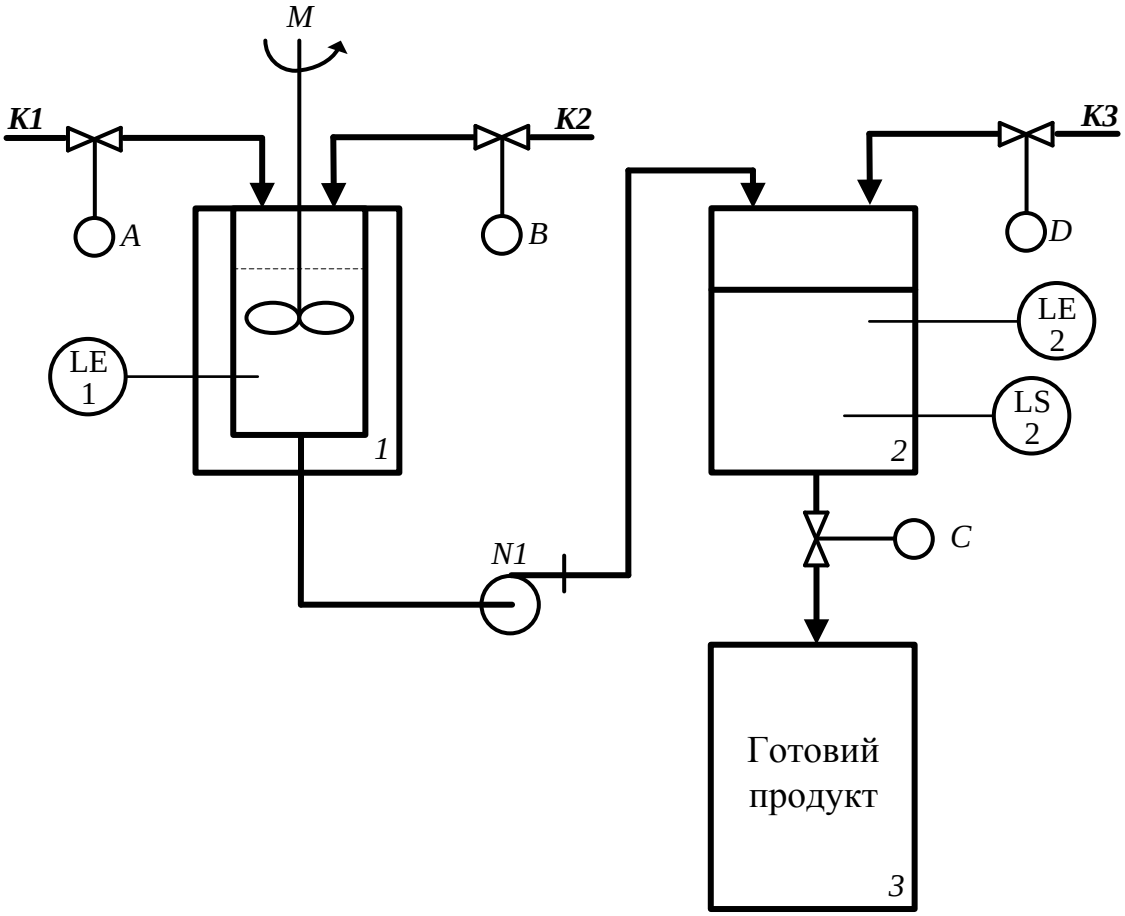


Рис. 2.1. Приклад АСК типового ОА

2) Виконати контрольне завдання до ЛР №2.

ЗАУВАЖЕННЯ. Давачі `LS` – булеві (тип логічний/булевий, реле).

Давачі `LE` – НЕ булеві (тип цілий або дійсний).

Задавачі – `L1_SET, L2_SET`

2.1 Постановка (скорегована) задачі розробки програми АСК

Розробити АСК із наступними вимогами:

1) Доповнити словники змінних моделі ОА та АСК;

2) Доповнити програмну модель ОА мовою ST давачами;

3) Доповнити керуючу програму мовами SFC/ST із додаванням зворотних зв′язків (давачів).

Структура віртуального лабораторного стенду №2

Лабораторний стенд складається з елементів, що відповідають ЛР№1.

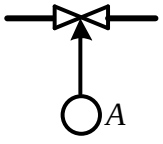
2.2 Теоретичні відомості

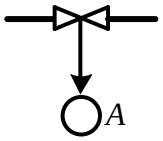
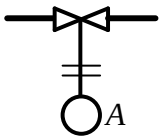
В ході виконання завдання треба враховувати наступні зауваження:

- діапазон зміни стану позиційних клапанів – 0-100%.
- в завданнях наведенні приблизні часові рамки виконання процесу. Для моделювання процесів дозволяється використати довільний час. Бажано зменшити до декількох, або десятків секунд.
- при включенні системи всі клапани повинні бути закриті.
- графік $T_з(t)$ відповідає заданій (уставка) температурі (програмний задавач), а не поточній температурі, що зчитується з давача ТЕ.

В технологічних схемах використовуються три типи клапанів.

В межах ЛР приймемо, що клапани зі стрілками на лінії від виконавчого механізму (ВМ) керуються логічними сигналами TRUE/FALSE себто клапан має два положення: відкритий або закритий. Клапан із рисками на лінії від ВМ керується аналоговим сигналом (від 0 до 100%).

	Виконавчий пристрій у складі приводу та клапану, який закривається у випадку втрати електричного живлення (нормально закритий клапан)

	Виконавчий пристрій у складі приводу та клапану, який відкривається у випадку втрати електричного живлення (нормально відкритий клапан)
	Виконавчий пристрій у складі приводу та розподіленого клапану, який залишається в тому самому положенні у випадку втрати електричного живлення (позиційний клапан)

2.3 Приклад виконання роботи

Розглянемо програму на мові SFC, яка реалізує алгоритм приведеної нижче послідовності дій АСК:

1. При натисненні кнопки «**СТАРТ**» розпочати процес, при цьому відкрити клапан **A** і завантажувати компонент *K1* в ємність *1* доки значення на давачі **LE1** не досягне заданої уставки **L1_SET**.
2. По закінченні завантаження компонента *K1*, при працюючій мішалці **M** розпочати завантаження компонента *K2*, відкривши клапан **B**.
3. Завантаження здійснювати до оновленої уставки **L1_SET**.
4. По закінченні завантаження компонента *K2* суміш продовжувати перемішувати **ще протягом 10 c**.
5. Потім суміш перекачати в ємність 2, включивши насос **N1**. Про спорожнення ємності *1* свідчить нульове значення на давачі **LE1**.
6. В ємність *2* завантажувати компонент *K3*, відкривши клапан **D** доки значення на давачі **LE2** не досягне заданої уставки **L2_SET**.
7. При досягненні рівня в ємності *2* значення 10 спрацьовує **LS2**.
8. Суміш повинна витримуватись протягом 15 c, після чого відкрити клапан **C** та вивантажити готовий продукт з ємності *2* в ємність *3*. Про спорожнення ємності *2* свідчить нульове значення на давачі **LE2**.
9. Підготувати лінію для приготування нової партії продукту.

1) Доповнити словник змінних АСК згідно таблиці 1.6.

Таблиця 1.6 – Доповнення словнику змінних АСК

Назва змінної	Тип	Опис
L1_SET	WORD	Змінна уставки (заданого) рівня у ємності 1
L2_SET	WORD	Змінна уставки (заданого) рівня у ємності 2
LE1	WORD	Змінна значення з давача рівня ємності 1
LE2	WORD	Змінна значення з давача рівня ємності 2
levelA	WORD	Приклад уставки рівня 1-го компоненту
levelB	WORD	Приклад уставки рівня 2-го компоненту
levelC	WORD	Приклад уставки рівня 3-го компоненту
LS2	BOOL	Логічний давач фіксованого рівня ємності 2

2) Взяти за основу проект з ЛР №1. Перейменувати ФБ lab1_fb на lab2_fb.

3) **Додати** до розділу декларування ФБ **Model_OA** (у блок вихідних змінних VAR_OUTPUT) наступний код:

```
VAR_OUTPUT
     LE1, LE2: WORD:=0;
     LS2: BOOL:=FALSE;
END_VAR
```

4) У ФБ **Init додати** в кінець наступний код (реалізація давачів на ОА):

```
LE1:=REAL_TO_WORD(h1); LE2:= REAL_TO_WORD(h2);
LS2:=LE2 >= 10; // давач спрацює при досягненні рівня 10
```

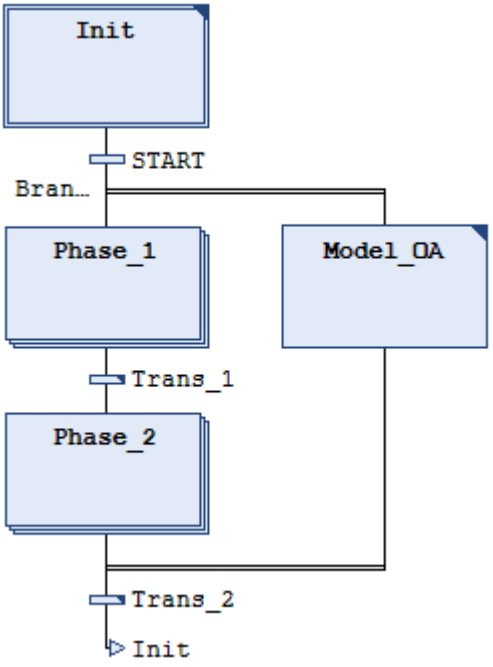
5) **Додати** до розділу декларування ФБ **lab2_fb** (у блок вхідних змінних VAR_INPUT, вихідних VAR_OUTPUT й змінних VAR) наступний код:

```
VAR_INPUT
     levelA: WORD;
     levelB: WORD;
     levelC: WORD;
END_VAR

VAR_OUTPUT
     H1, H2: WORD;
END_VAR
VAR
```

```
        L1_SET, L2_SET: WORD:=0;
        HS2: BOOL:=FALSE;
END_VAR
```

6) В розділі реалізації потрібно модифікувати код (рис. 2.2-2.4).

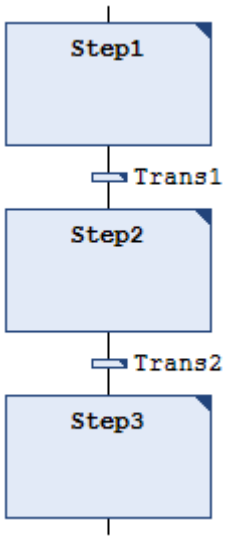


Елемент	*Код мовою ST*

Init	START:=TRUE; v_in1:=FALSE;
	v_in2:=FALSE; v_in3:=FALSE;
	v_out1:=FALSE;
	m:=FALSE; p:=FALSE;
	L1_SET:=0; L2_SET:=0;

START	START
Phase_1	див. рис. 2.3
Trans_1	Step3.t>=T#10S; // MIXING…
Phase_2	див. рис. 2.4
Trans_2	H2<=L2_SET; //L2_SET=0
Model_OA	model_fb(valve_in1:=v_in1, valve_in2:=v_in2, valve_in3:=v_in3, valve_out1:=v_out1, mixer:=m, pump:=p; LE1=>H1; LE2=>H2; LS2=>HS2);

Рис. 2.2. Схема програми керування АСК мовами SFC та ST

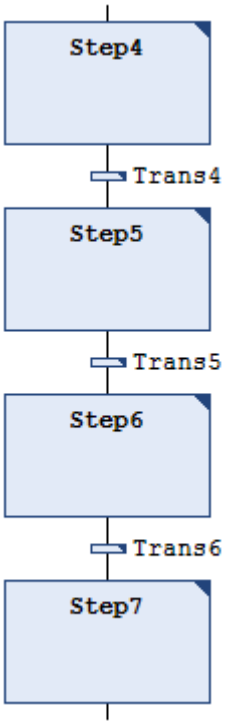


Елемент	*Код мовою ST*
Step1	v_in1:=TRUE; L1_SET:=levelA;
Trans_1	H1>=L1_SET;

Step2	v_in1:=FALSE; v_in2:=TRUE;
	L1_SET:=levelA+levelB;
	m:=TRUE;

Trans_2	H1>=L1_SET;
Step3	v_in2:=FALSE;

Рис. 2.3. Модифікована реалізація макро-елемента Phase_1



Елемент	*Код мовою ST*
Step4	`m:=FALSE; n1:=TRUE;` `L1_SET:=0;`
`Trans_4`	`n1 AND H1<=L1_SET;`
Step5	`n1:=FALSE; v_in3:=TRUE;` `L2_SET:=levelA+levelB+levelC;`
`Trans_5`	`H2>=L2_SET;`
Step6	`v_in3:=FALSE;`
`Trans_6`	`H1S`
Step7	`v_out1:=TRUE; L2_SET:=0;`

Рис. 2.4. Модифікована реалізація макро-елемента Phase_2

7) В головної програмі викликаємо ФБ **lab2_fb**:

Декларація:

```
PROGRAM PLC_PRG
VAR
     pba_fb: lab2_fb;
END_VAR
```

Реалізація:

```
pba_fb(levelA:=3.0,levelB:=5.0, levelC:=4.0);
```

2.4 Контрольні запитання

1) В чому полягає відмінність Давача (датчика) від Задавача?

2) Яку функцію виконує пристрій LE в моделі АСК?

3) У чому полягає відмінність LE та LS?

4) За якої умови робота АСК переходить від Фази 1 до Фази 2?

5) Що таке макро-елемент?

6) Які переваги надають пристрої зворотного зв′язку для АСК?

7) Поясніть відмінність нових графіків із попередньою роботою.

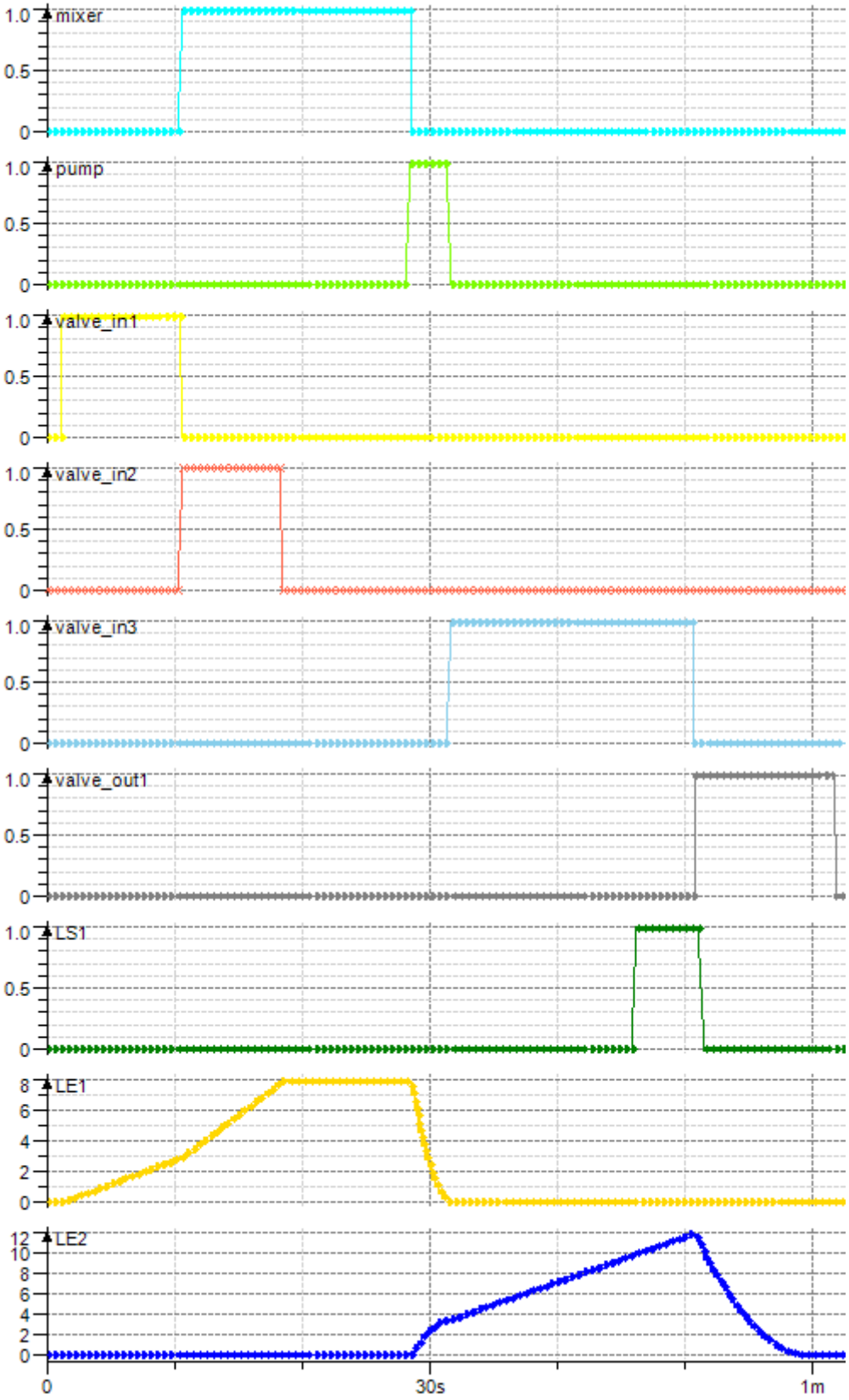


Рис. 2.5. Часова діаграма роботи АСК зі зворотними зв′язками

2.5 Контрольне завдання

1) Додати до вихідного проекту новий об'єкт ***Додаток*** з назвою відповідно до шаблону **PIP_LR02.** Встановити його активним.

2) За власним варіантом у файлі «**ПрІС. ЛР. БІЗ.pdf**» віднайти та виконати індивідуальне завдання до ЛР №2.

- записати алгоритм функціонування АСК та доповнити змінних й параметрів АСК.
- модифікувати програму керування технологічною схемою.

Результати ЛР подати у вигляді графіків вхідних та вихідних сигналів, побудованих в **CoDeSys.Trace**.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

3 ЛАБОРАТОРНА РОБОТА № 3

Тема: Створення скрипту імітатора об'єкта керування

Мета: Вивчення основних положень та принципів розробки програмної моделі (скрипту) об'єкту автоматизації (ОА) на мові **VBScript**.

Завдання:

1) Ознайомитись із базовими положеннями та синтаксисом мови програмування **VBScript**.

2) Засвоїти методику та послідовність розробки віртуального ОА на прикладі ОА із ЛР №2 (рис. 2.1).

3) Виконати контрольне завдання до ЛР №3.

3.1 Постановка задачі розробки програмної моделі об'єкту

Розробити програму що імітує життєдіяльність ОА:

1) Використати програмні напрацювання імітації ОА з ЛР №1-2;

2) На мові **VBScript** розробити модель ОА у вигляді скрипт-файлу;

3) Модифікувати програмну модель ОК до типу ОА наступним чином: змінні/компоненти моделі ОК «прив'язуються» (у скрипті) до регістрів ТЗА (керуючі сигнали в сегменті Coils; а за(давачі) в сегменті DI або через регістри зберігання) із підтримкою MODBUS-функцій;

4) Результуючий скрипт моделі ОА завантажити до fmPLCs;

5) Виконати запуск емуляції роботи ВОА.

Лабораторний стенд на базі ПК складається із наступних елементів:

1) Середовище програмування CoDeSys;

2) ВК – **CoDeSys Control Win**.

3) **ВОА – автоматичний симулятор** ОК + ТЗА – **fmPLCs + скрипт**;

4) Найпростіший редактор/транслятор мови програмування BASIC;

Структура стенду наведена на рис. 3.1.

<u>ЗАУВАЖЕННЯ</u>. Симулятор ОА на стенді відсутній. Тож він розробляється студентом-виконавцем.

Функціонування стенду №3 наступне. Скрипт симулятора ОА підключається до віртуального стенду шляхом завантаження скрипту, реалізованого мовою **VBScript**, до інтерпретатора моделі fmPLCs.

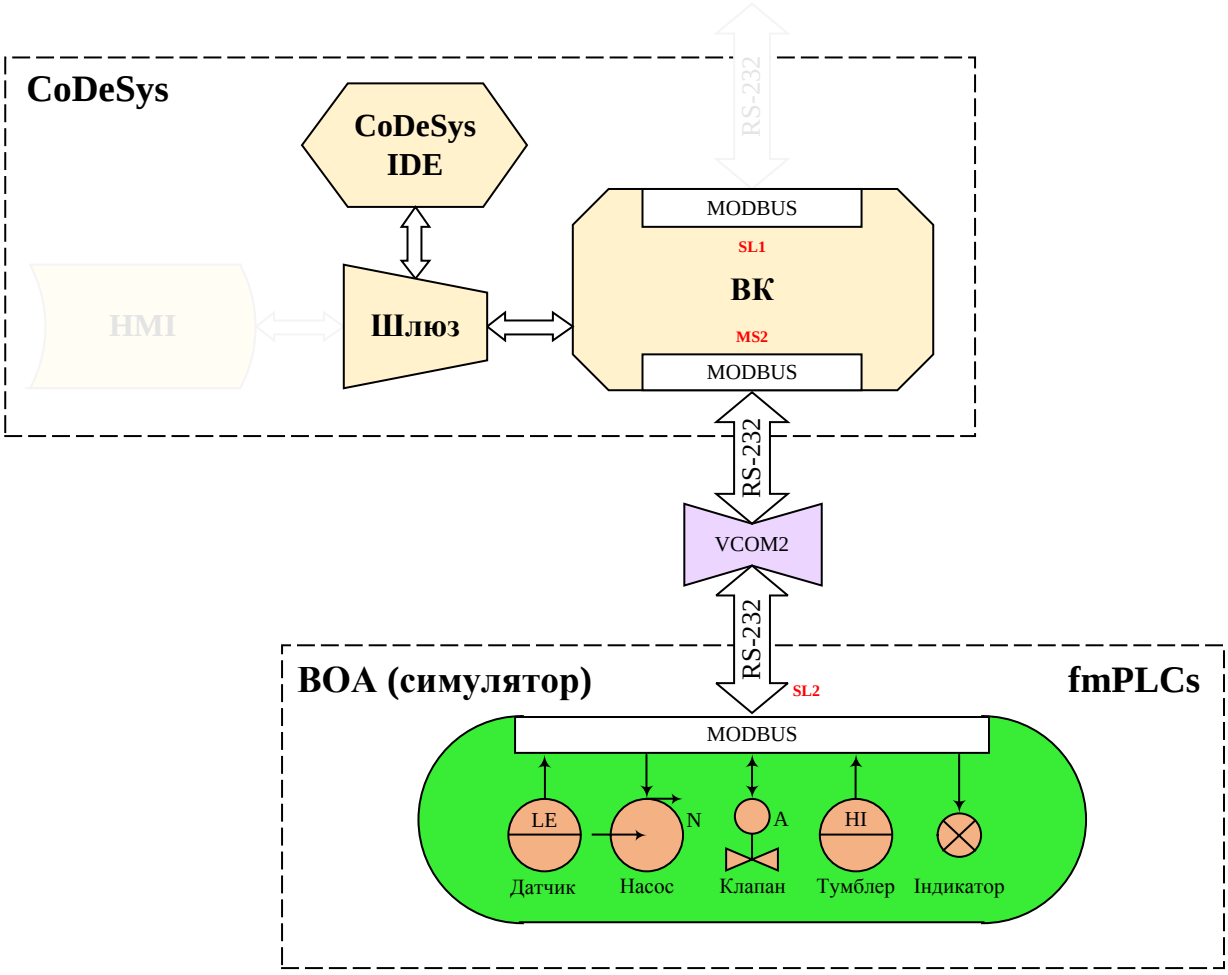


Рис. 3.1. Структура віртуального лабораторного стенду №4

Програмна модель ВОА є звичайним текстовим файлом (*.vbs), що містить скрипт – прикладу програму на мові **VBScript**. Скрипт формує поведінку (життєдіяльність) ОА **протягом одного такту** свого виконання.

Скрипт – це дискретна модель із тактом дискретизації. Час такту залежить від ПК, де відбувається його виконання інтерпретатором fmPLCs.

Інтерпретатор fmPLCs виконує скрипт з першого рядка до останнього за один такт, після чого виконання починається знову з першого рядка скрипту. Робота скрипту – це динамічна зміна стану ОА із плином часу.

На рівні ВОА у якості задавачів можуть виступати кнопки, тумблери, ручки, потенціометри тощо.

Скрипт ВОА складається із таких **послідовних програмних блоків**:

1) **Зчитування** даних із ТЗА (за/давачі та команди від ВК);

2) **Алгоритм** функціонування, одного такту життєдіяльності ОА;

3) **Запис** оновленого стану давачів ОА до ТЗА.

Перший блок має обов'язково містити такий метод MODBUS:

GetRegisterValue.

Третій блок має обов'язково містити такий метод MODBUS:

SetRegisterValue.

Детальна документація щодо використання цих методів наведена в окремому документі-інструкції з користування інтерпретатором fmPLCs на лабораторному віртуальному стенді.

3.2 Теоретичні відомості про синтаксис мови VBScripts

Оголошення змінних

```
Dim Var1, Var2
```

Варіанти і підтипи

У **VBScript** є лише один тип даних, названий **варіантом** (Variant). Варіанти можуть зберігати дані в підтипах даних. У табл. 3.1 описані основні підтипи, підтримувані **VBScript**.

В рамках даної ЛР використовуються лише наступні типи даних: **Boolean** (у CoDeSys це BOOL) та **Integer** (у CoDeSys це WORD).

Присвоєння значень: `Ім'я_змінної = значення`

Оголошення масивів: `Dim States (50)`

Таблиця 3.1 – Типи даних VBS

Підтип	Опис	Розмір
Boolean	**True** або **False**	1 біт (**BOOL**)
Byte	Ціле число від 0 до 255	1 байт
Integer	Ціле від -32768 до 32767	2 байти (**WORD**)
Long	Ціле від -2147483648 до 2147483647	4 байти
Single	Дійсне число звичайної точності	4 байти (**REAL**, **FLOAT**)
Double	Дійсне число подвійної точності	8 байт
String	Строкова змінна	N байт

Ця конструкція створює масив із 51 елемента. Перший елемент має нульовий номер, а число в дужках – номер останнього елемента масиву.

У **VBScript** регістр не має значення (на відміну від JavaScript і XML).

Все, що слідує за апострофом до кінця рядка, є **коментарем**.

Умовні вирази

У **VBScript** є дві форми умовних виразів:

1) **If ... Then ... Else** 2) **Select ... Case**

Конструкція **If ... Then ... Else** використовується для перевірки умови, яке може виявитися істинним або хибним, і потім, в залежності від результату порівняння, для виконання одного рядка коду. Кілька виразів поміщають між операторами **Then** та **End If**.

Конструкція **If ... Then ... Else If** перевіряє всі умови або до тих пір, поки не буде знайдено виконане, або до виразу **Else**.

Конструкція **Select ... Case** є альтернативною блоку **If ... Then ... Else,** коли з'являються складні умови. Вона добре підходить для ситуації, в якій є кілька можливих значень для перевірочної умови (вибір).

Синтаксис виглядає так:

```
Select Case умова
    Case значення
    Case значення
  Case Else
End Select
```

Оператори циклу

VBScript підтримує чотири форми циклів:

1) **For ... Next** 2) **For Each ... Next**

2) **Do ... Loop** 4) **While ... WEnd**

Структура **For ... Next** може містити необов'язковий параметр **Step,** що задає крок зміни лічильника циклу (може бути меншим 0).

Приклад:

```
For counter = 1 to 12
result = 5 * counter
Next counter
```

Конструкція **For Each ... Next** обробляє кожен елемент масиву.

Конструкція **Do ... Loop** повторює виконання обраного блоку коду до виконання заданої умови. Два типи цієї структури наведені далі. Це **Do ... While** і **Do ... Until**.

Структура циклу **Do,** що містить в кінці ключове слово **While,** виконується доти, доки перевірочна умова істина. Можна перевіряти цю умову на початку чи в кінці циклу.

Структура циклу **Do,** що містить в кінці ключове слово **Until,** виконується доти, доки перевірочна умова хибна. Як і в структурі **Do ... While,** дозволяється ставити умову на початку або в кінці циклу.

Структура **While ... WEnd** повторює виконання обраного блоку коду доти, доки перевірочна умова виконується.

Функції та процедури

Структура підпрограми (процедури):

```
Sub Ім'я_підпрограми (параметри)
Код підпрограми
End Sub
```

Функція має аналогічну структуру:

```
Function Ім'я_функції (параметри)
Код функції
End Function
```

Мова **VBScripts** також підтримує стандартні математичні функції, перелік яких наведений в окремому довідниковому документі.

3.3 Приклад виконання роботи

Розробка програмних моделей ВОА базується на використанні ключових знань, умінь та досвіду, що були отримані протягом вивчення навчальної дисципліни «Моделювання процесів і систем».

Потрібно виконати розробку моделі ОА (рис. 2.1) із наявними вимірювальними та виконавчими елементами. Обов'язково необхідно в скрипті ОА навести математичні залежності інтегруючих (накопичувальних) фізичних величин, наприклад, для **Температури** це ***Кількість тепла***, для **Рівня рідини** – її ***Об'єм*** і т.д.

Одиниці виміру змінних моделі ОА: рівень – дециметр [дм]; об'єм – літр [л=дм^3], тона [т = м^3]; керуючі команди – булевий [1/0].

На базі словників змінних з ЛР №1-2 потрібно **оголосити словник змінних** ОА (ОК й ТЗА) та їх розміщення в регістрах fmPLCs (табл. 3.2).

ЗАУВАЖЕННЯ. Всі компоненти моделі ОА за цим прикладом розміщено у одному сегменті «**Holding Register**» (у fmPLCs він 3-й).

Далі треба сформувати таблицю зі списком параметрів ємностей ОА та ТЗА із зазначенням номінальних значень (табл. 3.3).

Таблиця 3.2 – Змінні ВОА

Змінна	**Зміщення**		**Пояснення**
Offset=0; (R) – до ВК, (W) – від ВК			**Offset** – зміщення регістрів в пам'яті fmPLCs
q1	offset+0	(R)	поточний об'єм компоненту у ємності 1 [л]
q2	offset+1	(R)	поточний об'єм компоненту у ємності 2 [л]
LE1	offset+2	(R)	поточний рівень компоненту у ємності 1 [дм]
LE2	offset+3	(R)	поточний рівень компоненту у ємності 2 [дм]
overflow1	offset+4	(R)	контроль ємності 1 – переповнена, 0 – ні [1/0]
overflow2	offset+5	(R)	контроль ємності 2 – переповнена, 0 – ні [1/0]
valve_in1	offset+6	(W)	сигнал закр./відкрит. впускного клапану **A** [1/0]
valve_in2	offset+7	(W)	сигнал закр./відкрит. впускного клапану **B** [1/0]
valve_in3	offset+8	(W)	сигнал закр./відкрит. впускного клапану **D** [1/0]
valve_out1	offset+9	(W)	сигнал закр./відкрит. випускного клапану **C** [1/0]

pump	offset+10	(W)	сигнал вмикання/вимикання насосу **N1** [1/0]
mixer	offset+11	(W)	сигнал вмикання/вимикання мішалки **M** [1/0]
LE1_max	offset+12	(R)	макс. рівень компоненту у ємності 1 [дм]
LE2_max	offset+13	(R)	макс. рівень компоненту у ємності 2 [дм]

Таблиця 3.3 – Параметри ВОА

Параметр	Ном.	Пояснення
r_tank1	10	радіус циліндричної ємності 1 [дм]
r_tank2	15	радіус циліндричної ємності 2 [дм]
q_max1	10000	максимальний об'єм ємності 1 [л]
q_max2	15000	максимальний об'єм ємності 2 [л]
r_hole1	0.5	радіус сточного отвору ємності 1 [дм]
r_hole2	0.5	радіус сточного отвору ємності 2 [дм]
q_in1	100	макс. швидкість притоку через клапан **A** [л/сек.]
q_in2	100	макс. швидкість притоку через клапан **B** [л/сек.]
q_in3	100	макс. швидкість притоку через клапан **D** [л/сек.]
q_out1		поточна швидкість стоку через отвір ємн. 1 [л/сек.]
q_out2		поточна швидкість стоку через клапан **C** [л/сек.]
q_in4		поточна швидкість притоку через насос **N1** [л/сек.]

Параметри, що в попередній таблиці не ініціалізовано значеннями за замовчуванням обраховуватимуться в тілі скрипта моделі ОА.

Допоміжні параметри моделювання ОА сґрупувати у табл. 3.4.

Таблиця 3.4 – Допоміжні параметри ВОА

Параметр	Ном.	Пояснення
dt	0.1	тривалість одного такту «життя» ОА [сек.]
g	98	прискорнення вільного падіння [дм/сек.^{2}]
pi	3.1415	константа π

Інтерпретатор fmPLCs підтримує математичні функції мови **VBScripts**, наприклад, SQR() – функція обрахунку квадратного кореня $\sqrt{x}$.

На цьому **оголошення** змінних та параметрів ОА завершено.

Далі потрібно переходити до програмного блоку **зчитування** даних із ТЗА, а саме зчитування поточного об'єму компонентів у ємностях:

```
q1 = GetRegisterValue(3, offset)
q2 = GetRegisterValue(3, offset+1)
```

Зчитування **командного стану керуючих сигналів** з ВК до виконавчих пристроїв ОА:

```
valve_in1 = GetRegisterValue(3, offset + 6)
valve_in2 = GetRegisterValue(3, offset + 7)
valve_in3 = GetRegisterValue(3, offset + 8)
valve_out1= GetRegisterValue(3, offset + 9)
pump = GetRegisterValue(3, offset + 10)
mixer= GetRegisterValue(3, offset + 11)
```

Далі розробити програмний блок **логіки функціонування**, одного такту життєдіяльності ОА.

Поточна швидкість подачі компонентів у ємності:

```
q_in1 = q_in1*valve_in1
q_in2 = q_in2*valve_in2
q_in3 = q_in3*valve_in3
```

Поточний рівень компонентів у ємностях:

```
LE1 = q1/(pi*r_tank1*r_tank1)
LE2 = q2/(pi*r_tank2*r_tank2)
```

Максимальні значення рівнів компоненту в ємностях:

```
LE1_max = q_max1/(pi*r_tank1*r_tank1)
LE2_max = q_max2/(pi*r_tank2*r_tank2)
```

Перевірка на порожність ємності 1:

```
if (q1 = 0) then
     q_out1 = 0
else 'Поточна швидкість витоку компоненту із ємності 1
q_out1 = Sqr(2 * g * LE1) * pi * r_hole1 * r_hole1 * pump
     q_in4 = q_out1
end if
```

Перевірка на порожність ємності 2:

```
if (q2 = 0) then
     q_out2 = 0
else 'Поточна швидкість витоку компоненту із ємності 2
     q_out2 = Sqr(2 * g * LE2) * pi * r_hole2 * r_hole2 *
              valve_out1
end if
```

Накопичення компоненту у ємностях:

```
dq1 = (q_in1 + q_in2 - q_out1)*dt
q1 = q1 + dq1
dq2 = (q_in3 + q_in4 - q_out2)*dt
q2 = q2 + dq2
```

Друга перевірка на порожність ємності 1 (для ємності 2 аналогічно):

```
if (q1 <= 0) then
   q1 = 0
   LE1 = 0
end if
```

Перевірка на переповнення ємності 1 (для ємності 2 аналогічно):

```
if (q1 >= q_max1) then
   q1 = q_max1
   LE1 = LE1_max
   overflow1 = 1
else
   overflow1 = 0
end if
```

І нарешті, останній програмний блок **запису** стану давачів технологічних параметрів ОА, а саме давачів рівню та контролю переповнення:

```
SetRegisterValue 3, offset,   Int(q1)
SetRegisterValue 3, offset+1, Int(q2)
SetRegisterValue 3, offset+2, Int(LE1)
SetRegisterValue 3, offset+3, Int(LE2)
SetRegisterValue 3, offset+4, overflow1
SetRegisterValue 3, offset+5, overflow2
```

Для підвищення точності, замість функції відкидання дробової частини числа, можна застосувати функцію округлення числа.

Інформація щодо обрахованих параметрів ОА:

```
SetRegisterValue 3, offset+12, Int(LE1_max)
SetRegisterValue 3, offset+13, Int(LE2_max)
```

Також потрібно додати до скрипту моделі ОА функції виведення стану технологічних параметрів та керуючих сигналів у термінальне вікно fmPLCs, а саме, наприклад, рядок `AddDebugString "LE1=" +CStr(LE1).`

Наприкінці необхідно зібрати та допрацювати всі приведені вище фрагменти в один скрипт та отримати повноцінну програмну модель ОА.

Скріпт ВОА необхідно завантажити до fmPLCs у відповідності до рис. 84-б, обравши режим «**Training PLC simulation**». Якщо під час натискання

кнопки вибору шляху до файлу виникають проблеми, необхідно спробувати вручну записати шлях до файлу зі скриптом.

Далі виконати перевірку моделі ОА наступним чином:

1) В ручному режимі почергово вмикати виконавчі пристрої та у термінальному вікні спостерігати зміну стану давачів ОА;

2) Зберегти результати життєдіяльності ОА шляхом зняття скриншотів та супровідним пояснювальним текстом до них.

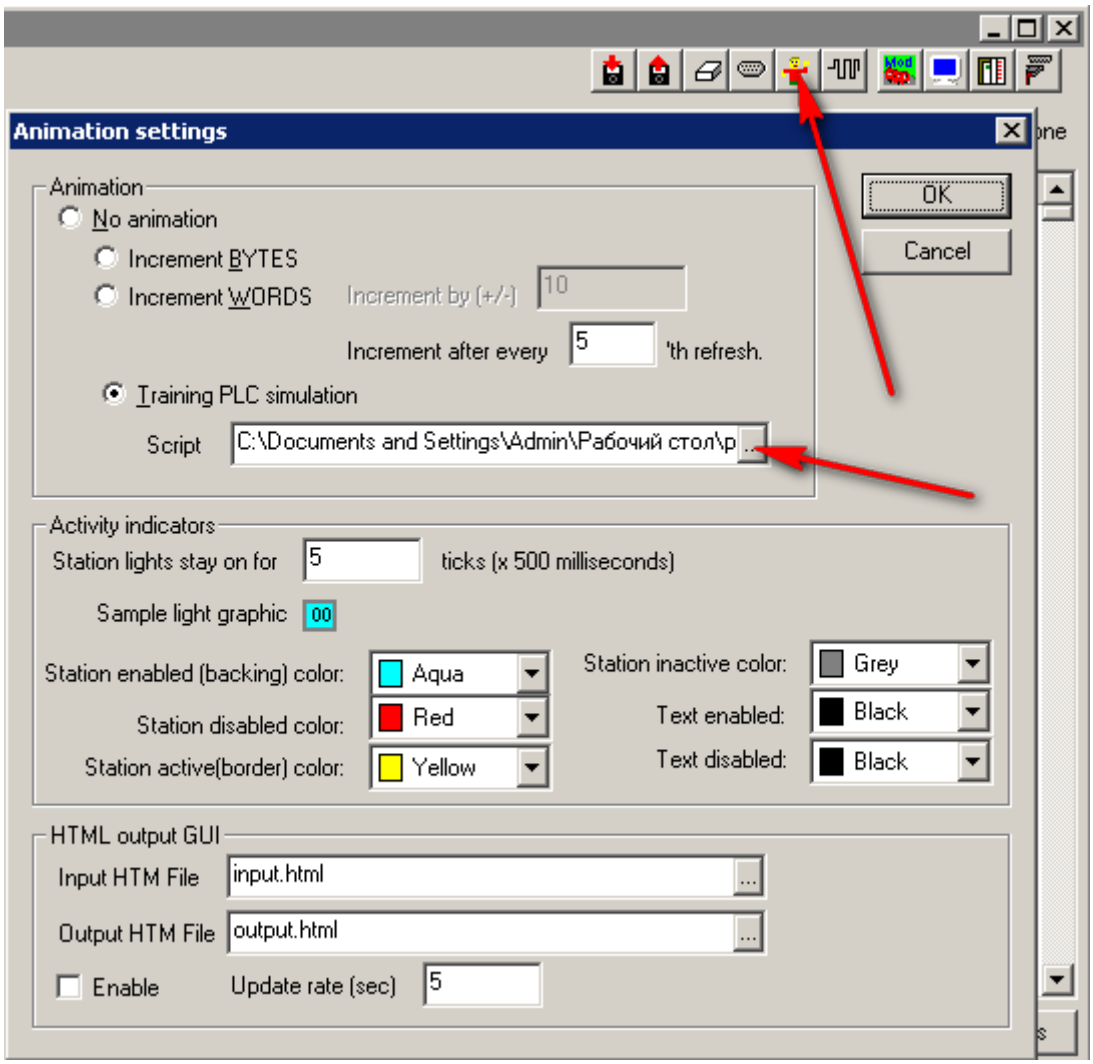


Рис. 3.2. Завантаження скрипту-моделі ОА до fmPLCs

ЗАУВАЖЕННЯ. Скрип програмної моделі ОА ні в якому разі не має містити: задавальні уставки; змінні, параметри та функції управління, керування, регулювання. Ціми елементами та задачами оперує ВК (або

зовнішній блок керування, електроавтоматики) й саме він формує керуючі команди до ВОА, а ВОА їх відпрацьовує відповідно до скрипту моделі ОА.

Забороняється використання циклів у скрипті ОА. Тактування виконувати із використанням зовнішнього циклу виклику скрипта програмою fmPLCs. Зміна стану ОА (наприклад, значення з давачів) синхронізується саме за цим тактуванням.

3.4 Контрольні запитання

1) Наведіть головні відмінності мов BASIC та VBScripts;
2) Якій структурі має підкорятись програмна модель ВОА?
3) Яким чином у скрипті моделі ВОА додати коментарі?
4) Що таке ОК, ОА та ВОА, у чому полягає їх відмінність?
5) Віртуальні давачі та задавачі. Наведіть приклади;
6) Лінеаризація даних з давачів, у чому суть?
7) Поясність концепцію ВОА та життєдіяльність віртуального ОА.

3.5 Контрольне завдання

1) Створити модель ОА «**IAxxNN-PIP03**» (xx, NN – див. п.1.6) у вигляді скрипт-файлу, використавши напрацювання ОА з ЛР №1-2.

2) Включити до скрипту моделі ОА: стани давачів, **інтегруючих фізичних величин ОА**, «природне» обмеження фізичних величин в ОА.

3) Врахувати форми устаткування ОА, діапазон зміни керуючих сигналів та погрішність давачів (за наявності у завданні).

4) Додати до скрипту ОА вивід параметрів у термінальне вікно.

5) Підключити ОА до fmPLCs (ВОА), запустити відповідні виконавчі елементи та перевірити динаміку роботи ОА.

Результати ЛР подати у вигляді скриншотів Логу програми fmPLCs.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

4 ЛАБОРАТОРНА РОБОТА № 4

Тема: Застосування протоколу MODBUS для комунікації ВК з ОК

Мета: Вивчення основних положень стандарту та можливостей промислового протоколу MODBUS на практичному прикладі АСК.

Завдання:

1) Ознайомитись із технологією віртуальних інтерфейсів на прикладі встановлення віртуального порту RS-232 на платформу розробника (ПК);

2) Засвоїти методику та послідовність конфігурування та налаштування протоколу MODBUS в середовищі CoDeSys;

3) Виконати контрольне завдання до ЛР №4.

4.1 Постановка задачі застосування протоколу MODBUS

Інтегрувати в проект та налаштувати комунікаційний протокол MODBUS для організації передачі даних між ВК та ВОА:

1) Використати за базу робочий вихідний проект із ЛР №2;

2) Перевірити наявність на ПК пари віртуальних портів RS-232;

3) Із проекту CoDeSys вилучити фрагменти коду (схему, ФБ), змінні, що відповідають за імітацію ОА;

4) **Налаштувати порти ВК на поточний *Додаток***: «**Device → Edit Object → PLC Settings → Application for I/O handling → PIP_LR04**».

5) Додати до проекту комунікаційний канал **MODBUS** та COM-порт;

6) Сформувати таблицю зі словником змінних АСК;

7) Налаштувати параметри передачі-прийому у відповідністю із таблицею змінних проекту. Сформувати таблицю регістрів **MODBUS**;

8) Створити **CoDeSys.Trace** для побудови динаміки зміни значень технологічних параметрів та керуючих команд АСК;

9) Скомпілювати модифікований проект та завантажити його до ВК;

10) Запустити ВОА та проімітувати роботу АСК наступним чином: вводити бінарні команди (наприклад, СТАРТ) в ВОА та спостерігати їх відпрацювання у ВК засобами **CoDeSys.Trace**. ВК у відповідь має формувати керуючі команди, які запишуться у відповідні регістри ВОА. Далі, змінюючи стан давачів (регістри ВОА), спостерігати реакцію (стан) ОА у **CoDeSys.Trace**.

Лабораторний стенд на базі ПК складається із наступних елементів:

1) Середовище програмування CoDeSys;

2) ВК – **CoDeSys Control Win**.

3) Віртуальний RS-232 – «**ELTIMA Virtual COM**» – VCOM;

4) Віртуальний об'єкт автоматизації (ВОА) – **ручний імітатор** ОК та ТЗА – програма «**Free Modbus PLC Simulator**» (fmPLCs);

Структура стенду наведена на рис. 4.1.

Якщо на ПК встановлено віртуальний COM-порт, необхідно обрати «**Мій комп'ютер → Властивості → Диспетчер пристроїв**» та знайти номери віртуальних COM-портів інтерфейсу ELTIMA. На рис. 4.1 наведений приклад коли номери віртуальних COM-портів є COM1 та COM2. Вони віртуально **з'єднані** (аналог фізичного з'єднання кабелем).

Функціонування стенду №4 наступне. Прикладна програма CoDeSys, завантажена у ВК, через інтерфейс віртуального COM1 (**MS**) передає команди протоколом **MODBUS**. Віртуальним каналом зв'язку команди передаються до віртуального COM2 (**SL**), який у свою чергу передає команди до ВОА, що надходять до виконавчих пристроїв АСК. В свою чергу ВОА повертає до ВК необхідні дані з давачів про поточний стан ОК.

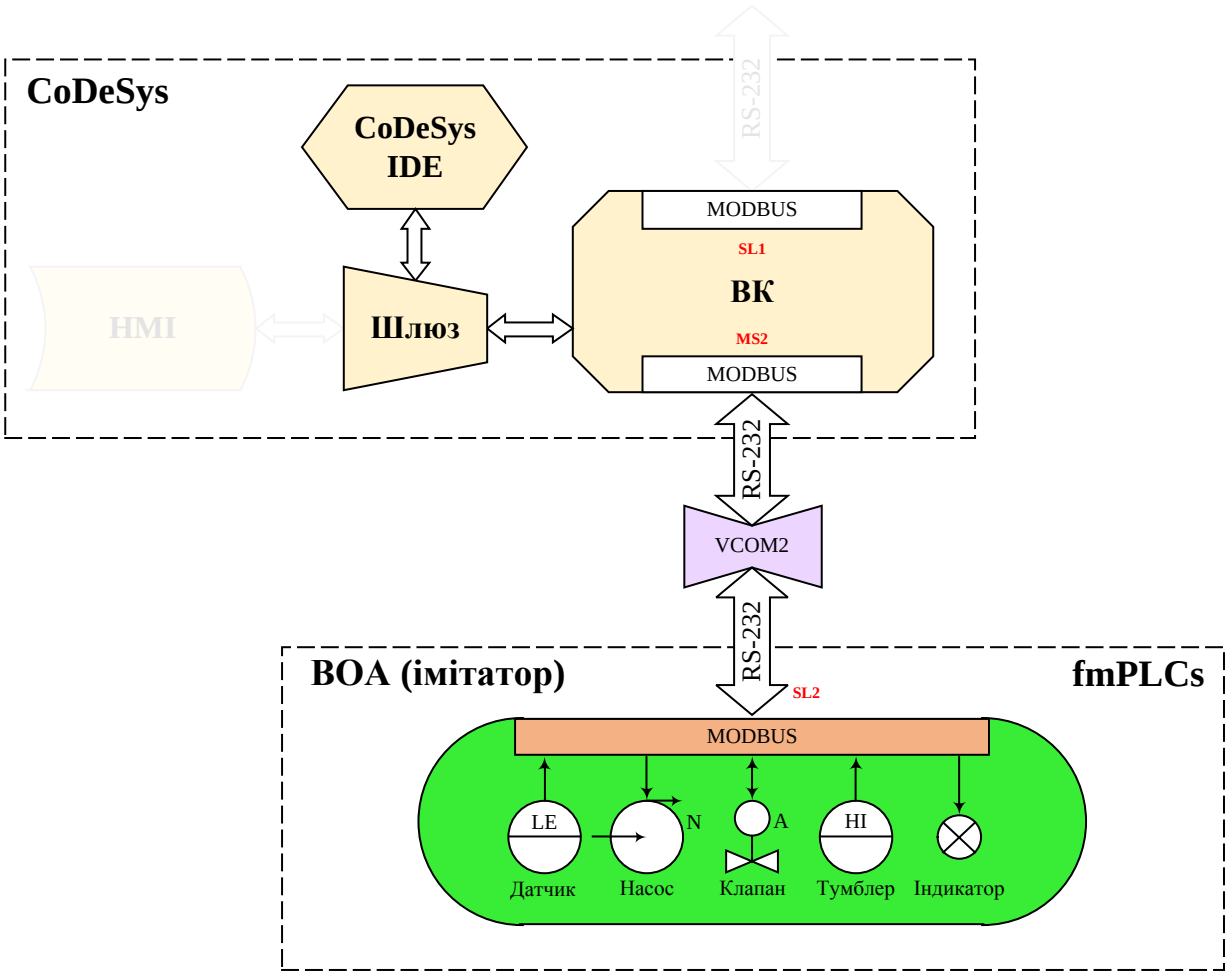


Рис. 4.1. Структура віртуального лабораторного стенду №4

4.2 Теоретичні відомості про стандарт MODBUS

Промисловий протокол **MODBUS** призначений для передачі команд/даних від Мастера (**MS**) до Слейву (**SL**) та отримання від **SL** відповідних відповідей/даних. **MODBUS** оперує даними які адресуються у форматі **Y**XXX(X), де **Y** – зона (**сегмент**, область) даних; XXX – зміщення відносно початку (000) зони; (X) – номер біту у байті (слові) даних.

MODBUS підтримує 4 типи даних:

1) **Discrete Inputs 1**XXX(X) – один біт, доступний тільки для зчитування дискретних **входів** ПЛК;

2) **Coils 0**XXX(X) – один біт, доступний для зчитування і запису (реле, котушка) логічних комірок (дискретних **виходів**) ПЛК;

3) **Input Registers 3**XXX(X) – 16-бітний беззнаковий або знаковий тип, доступний тільки для зчитування (сигнали з давачів, АЦП) вхідних регістрів ПЛК;

4) **Holding Registers 4**XXX(X) – 16-бітний беззнаковий або знаковий тип, доступний для зчитування і запису (змінні й параметри АСК) внутрішніх регістрів ПЛК.

Структура повідомлення протоколу **MODBUS** наведена на рис. 4.2.

Адреса SL	Код функції	ДАНІ	CRC16
1 байт	1 байт	N байт (<255)	2 байта

Рис. 4.2. Структура MS-команди MODBUS-пакету

Структура кадру (пакету, фрейму тощо) **MODBUS** включає:

1) **адреса** (веденого пристрою **SL**) – це адреса пристрою, якому адресується дане повідомлення протоколу **MODBUS**. Пристрої **SL** відповідають тільки на ті повідомлення, які адресовані саме їм. Відповідь починається з адреси **SL**. Адреса змінюється в межах 1 ... 247. Адреса 0 в повідомленні протоколу **MODBUS** зарезервована під широкомовні повідомлення, 248..255 – зарезервовані адреси.

2) **код функції** – 1 байт даних.

3) **дані** – це поле містить інформацію про функцію (команду), яку потрібно виконати, або дані, які **SL** посилає **MS** протоколом **MODBUS**.

4) **CRC16** (блок виявлення помилок) – контрольна сума, яка обраховується з усіх попередніх байт шляхом алгоритму циклічного зсуву і побітового виключення.

ЗАУВАЖЕННЯ. При зчитуванні протоколом **MODBUS** в одному повідомленні можна зчитувати значення дискретних й аналогових входів і виходів, розташованих підряд, можна задати адресу першого значення та їх кількість. **Дані зберігаються і зчитуються у двох байтах** (тип **WORD**).

Основні **стандартні функції протоколу MODBUS** за їх кодами (у десятковому і шістнадцятковому форматі) наступні:

• 1 (**0x01**) – зчитування декількох дискретних виходів (DO=Coils);

• 2 (**0x02**) – зчитування декількох дискретних входів (DI);

• 3 (**0x03**) – зчитув. декількох проміжних регістрів/аналог. виходів;

• 4 (**0x04**) – зчитування декількох аналогових входів.

• 5 (**0x05**) – запис значення одного дискретного виходу (Coil).

• 6 (**0x06**) – запис значення одного аналогового виходу або регістра.

Команда **MODBUS** складається з адреси **SL** і власне значення (2 байта). Нормальна відповідь – повтор запиту протоколу **MODBUS**.

• 15 (**0x0F**) – запис значень у кілька дискретних виходів (Coils).

• 16 (**0x10**) – запис значень декількох аналогових виходів/регістрів.

4.3 Приклад виконання роботи

1) Створити новий ***Додаток*** у проекті по аналогії з ЛР №2.

Потрібно розробити програму на мові SFC на прикладі АСК для ОА, який зображено на рис. 2.1.

Пригадаємо словник змінних АСК (табл. 4.1).

Таблиця 4.1 – Словник змінних АСК

Назва змінної	Тип	Опис
L1_SET	WORD	Змінна уставки (заданого) рівня у ємності 1
L2_SET	WORD	Змінна уставки (заданого) рівня у ємності 2
LE1	WORD	Змінна значення з давача рівня ємності 1
LE2	WORD	Змінна значення з давача рівня ємності 2
levelA	WORD	Приклад уставки рівня 1-го компоненту
levelB	WORD	Приклад уставки рівня 2-го компоненту
levelC	WORD	Приклад уставки рівня 3-го компоненту
A	BOOL	Сигнал керування клапаном **A**
B	BOOL	Сигнал керування клапаном **B**
C	BOOL	Сигнал керування клапаном **C**

D	BOOL	Сигнал керування клапаном **D**
M	BOOL	Сигнал керування двигуном мішалки **M**
P	BOOL	Сигнал керування насосом **N1**
START	BOOL	Команда вмикання усієї АСК на рівні ВОА

2) Для того, щоб можна було виконати емуляцію взаємозв'язку пристрою середовища CoDeSys (ВК) та пристрою-емулятора MODBUS (ВОА), треба, щоби в ПК був присутній COM-порт (інтерфейс RS-232). Якщо такий порт відсутній, необхідно встановити емулятор COM-порту, наприклад, **Eltima Virtual Serial Port Driver 6.0** (рис. 4.3).

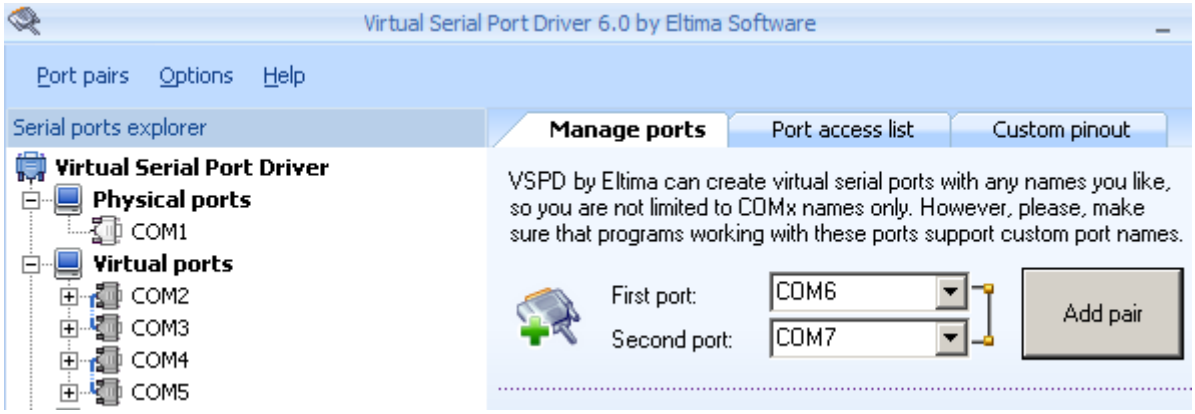


Рис. 4.3. Eltima Virtual Serial Port Driver 6.0

3) Далі необхідно додати до проекту CoDeSys новий пристрій MODBUS-каналу. Оскільки CoDeSys є мастером (**MS**) тобто ініціює транзакції (передачу команд) через послідовний інтерфейс RS-232 до ВОА **(SL)**, тому необхідно в проект додати гілку від **Device** – пристрій **Modbus_COM**, від нього **Modbus_Master_COM_Port** і в свою чергу від цього пристрою – **Modbus_Slave_COM**.

Для цього треба натиснути ПКМ на пристрої «**Device → Add Device**», вибрати «**Modbus → Modbus Serial Port → Modbus_COM**» і тиснем ЛКМ двічі (рис. 4.4). Аналогічно створити **Modbus_Master_COM_Port** як нащадка під пристроєм **Modbus_COM** і так само **Modbus_Slave_COM**.

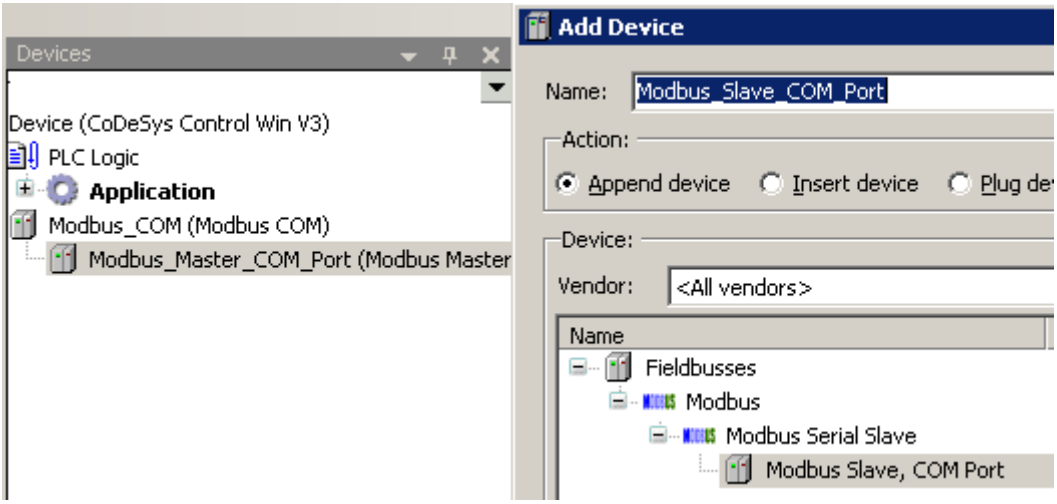


Рис. 4.4. Додавання нового MODBUS-пристрою

Наступним етапом треба виконати конфігурування цього пристрою **Modbus_COM**: на вкладці «**Modbus Serial Port Configuration»** у полі **COM Port** вказати номер порту (COM 1), який буде використовуватись ВК для трансакцій (рис. 4.5), інші поля залишити без змін.

4) Для налагодження зв'язку між ВК (MS) та пристроєм ВОА (SL) необхідно в проекті CoDeSys налаштувати MODBUS-канали: для отримання та відправки даних (провести так званий маппінг каналів на змінні).

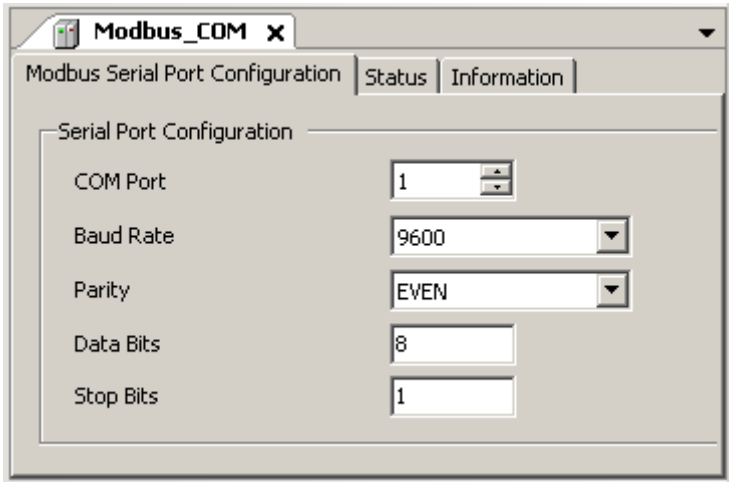


Рис. 4.5. Налаштування MODBUS-пристрою

<u>ЗАУВАЖЕННЯ</u>. *Додаток*, завантажений у ВК має співпадати із активним *Додатком* у проекті CoDeSys.

Маппінг – це процес встановлення відповідності між каналами введення/виведення та внутрішніми змінними програми. Табл. 4.2 показує, які змінні і по яким адресам отримують чи передають дані.

Таблиця 4.2 – Опис MODBUS-каналів у відповідності зі словником змінних АСК

Назва каналу	Тип операції (функції)	Зміщення	Сегмент
START	Read Discrete Inputs	16#000**0**	**1**
Executive	Write Multiple Coils	16#000**0**	**0**
LE1	Read Holding Registers	16#000**0**	**4**
LE2	Read Holding Registers	16#000**1**	**4**
L1_SET	Write Single Register	16#000**2**	**4**
L2_SET	Write Single Register	16#000**3**	**4**

Канал «START» є віртуальною кнопкою «**СТАРТ**» системи користувачем (а не ВК) на рівні ВОА безпосередньо через регістр у fmPLCs.

Адреси зміщень можуть повторюватися, наприклад, **16#0000** відповідає змінним **START** і **LE1.** Просто ці змінні зберігаються у різних областях (сегментах) пам'яті ВК. 16# – ширина розрядної сітки (**WORD**).

Щоб відкрити налаштування каналів (рис. 4.6, 4.7) необхідно зайти в налаштування пристрою **Modbus_Slave_COM_Port**, двічі клацнувши по ньому ЛКМ, перейти до «**Modbus Slave Channel»**, та натиснути на кнопку **Add Channel**. Після цього відкриється вікно, у якому необхідно задати ім'я, тип каналу та зміщення. За замовчуванням встановлено циклічне опитування адреси (змінної) через кожні 100 мс.

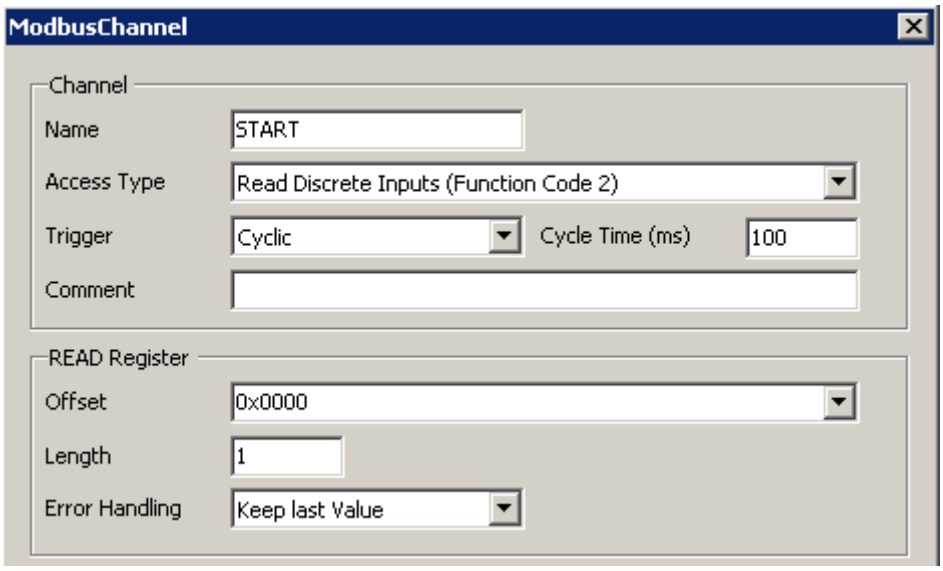


Рис. 4.6. Налаштування MODBUS-каналу для зчитування даних з мережі

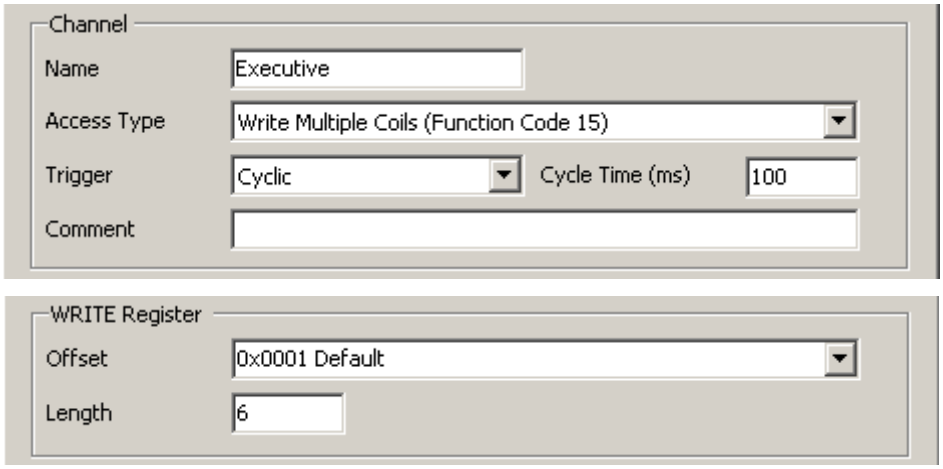


Рис. 4.7. Налаштування MODBUS-каналу для відправки даних в мережу

Тип каналу **Write Multiple Coils** (Function Code 15) дозволяє створити один канал з необхідною кількістю логічних змінних (**Discrete Outputs**), яка задається у полі **Length**. Усі змінні отримають своє значення зміщення починаючи з початкового, яке задається у полі **Offset**. Канал **Executive** обслуговує керуючі команди (булеві) 6-ти виконавчих пристроїв АСК (див. змінні A, B, C, D, M, N1 в табл. 4.1).

Далі необхідно встановити відповідність між MODBUS-каналами та внутрішніми змінними програми. Для цього у вікні налаштування пристрою **Slave**, на вкладці «**ModbusGenericSerialMaster I/O Mapping**» двічі

натиснути ЛКМ на відповідній стрічці в стовпці **Variable** та вибрати зі списку змінних необхідну змінну. У новіших (>3.4) версіях CoDeSys це вкладка «**ModbusGenericSerialSlave I/O Mapping**».

Далі необхідно виконати описані вище дії для всіх змінних які приймають участь в обміні даних MODBUS-мережею згідно табл. 4.2, де вказані відповідні їм регістри MODBUS.

Потім додати налаштування для усіх змінних на вкладці «**Modbus Slave Channel»** (рис. 4.8) та присвоїти їм відповідні змінні на вкладці «**ModbusGenericSerialMaster I/O Mapping»** (рис. 4.9).

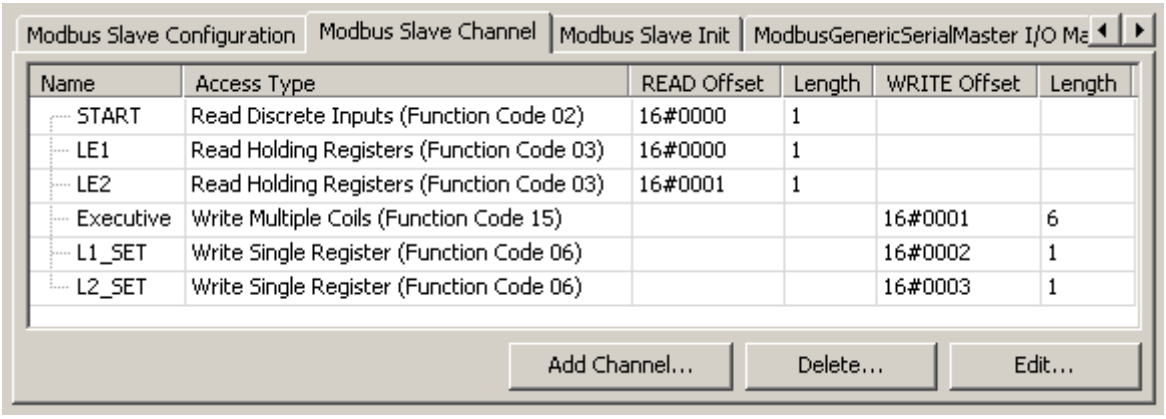


Рис. 4.8. Відображення Modbus Slave-каналів

Слід зазначити, що змінні L1_SET та L2_SET передаються до ВОА лише з метою індикації уставок рівня і ніяким чином не впливають на ОА.

На рис. 4.9 у колонці «**Type**» відображаються цілі типи даних без підтримки масивів. У новіших CoDeSys (ver.>3.4) типи у цій колонці відображаються масивом, що дозволяє без складнощів проводити маппінг змінних типу **REAL** лише обравши для неї **Length**=2 (рис. 4.8).

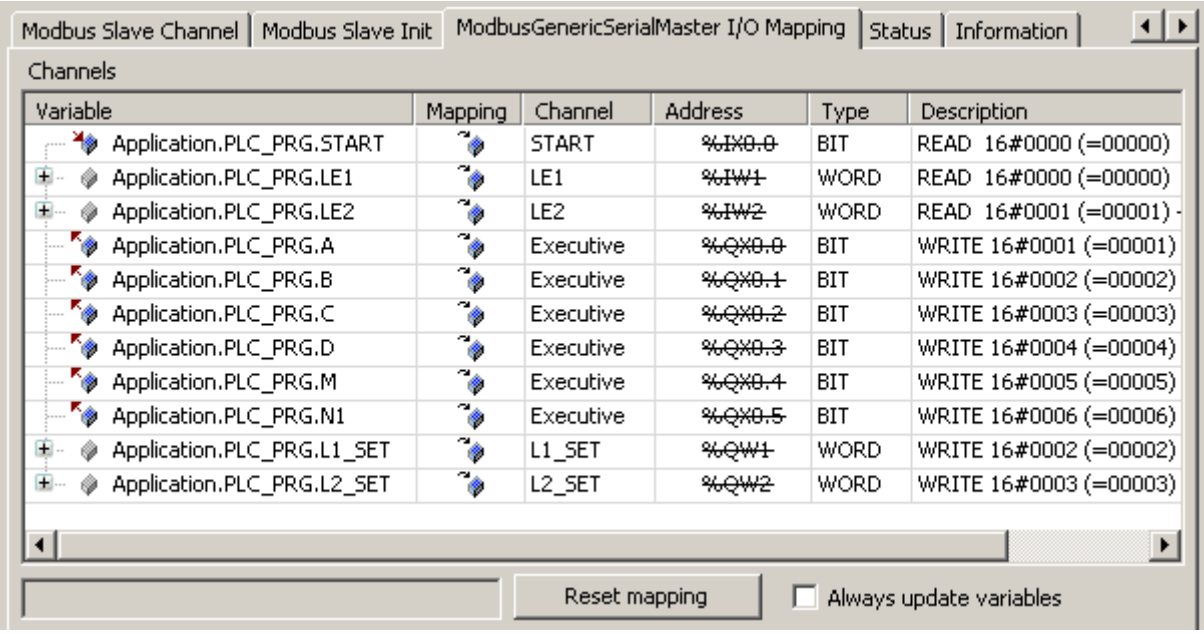


Рис. 4.9. Відображення маппінгу введення/виведення змінних

Після запуску програми на ВК і програми fmPLCs у Навігаторі проектів CoDeSys можна спостерігати факт успішної комунікації ВК та ручного імітатора ОА через MODBUS-мережу (зелені стрілки по колу на рис. 4.10).

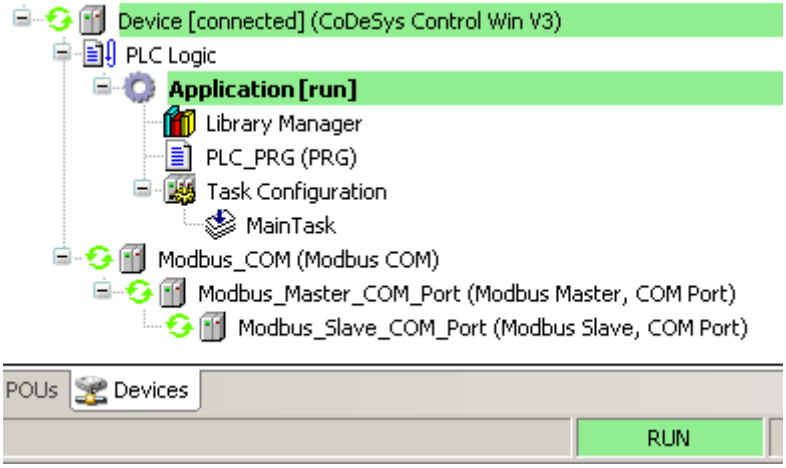


Рис. 4.10. Індикація успішної комунікації ВОА та ВК через MODBUS-RS232

5) Наступним кроком потрібно налаштувати ВОА (fmPLCs): в головному вікні програми fmPLCs у випадаючому списку **Port** вибрати **Modbus RS-232**, після цього натиснути на піктограму послідовного порту де обрати вільний COM-порт номер 2 (рис. 4.11).

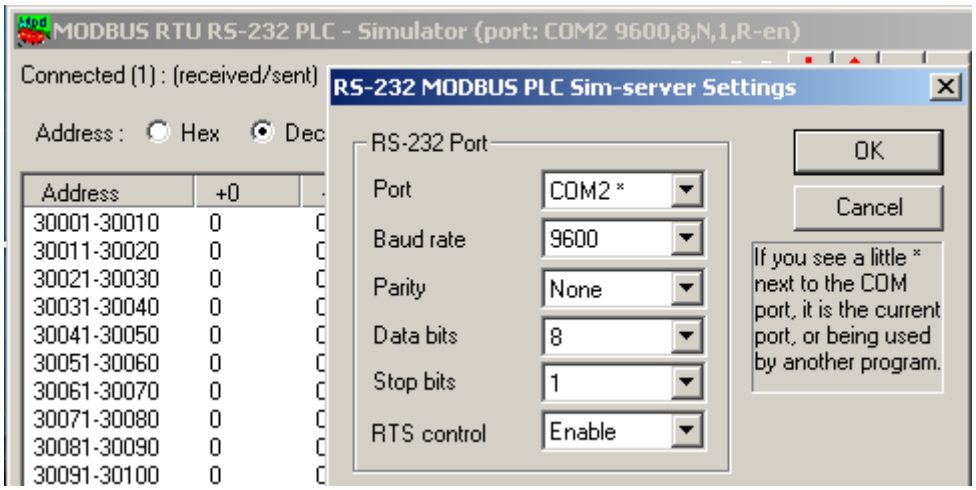


Рис. 4.11. Параметри передачі даних в пристрої ВОА

ЗАУВАЖЕННЯ. Оскільки на одній платформі ПК запускаються обидва пристрої (ВК та ВОА), необхідно щоб COM-порт вибраний для роботи ВК відрізнявся від COM-порту вибраного у ВОА, а також ці COM-порти мають бути **віртуально з'єднані**. В налаштуванні CoDeSys було обрано COM1, тому у ВОА необхідно обрати COM2, також перевіривши аби швидкість передачі даних в CoDeSys співпадала зі швидкістю у ВОА.

На рівні ВОА віртуально розміщено ОА (ємності) та технічні засоби автоматизації (ТЗА): клапани, насос, давачі й органи керування (у прикладі лише кнопка СТАРТ). Програма fmPLCs виконує функцію **імітації** поведінки ОА й дозволяє (в ЛР°№3) в ручному режимі моделювати ОА.

4.4 Алгоритм ручної імітації поведінки ОК

Крок 0. ВК постійно чекає на зовнішній сигнал «**СТАРТ**» системи.

Крок 1. Оператор запускає АСК кнопкою «**СТАРТ**». Для цього необхідно виконати запис логічної одиниці (подвійним натисканням ЛКМ) у сегменті «**Digital Inputs**» (на рис. 4.12 за стрілкою) до першого дискретного входу (на рис. 4.12 виділено кружком) у відповідності до проведеного раніше маппінгу змінних (рис. 4.10) з адресою **1**0001 у fmPLCs.

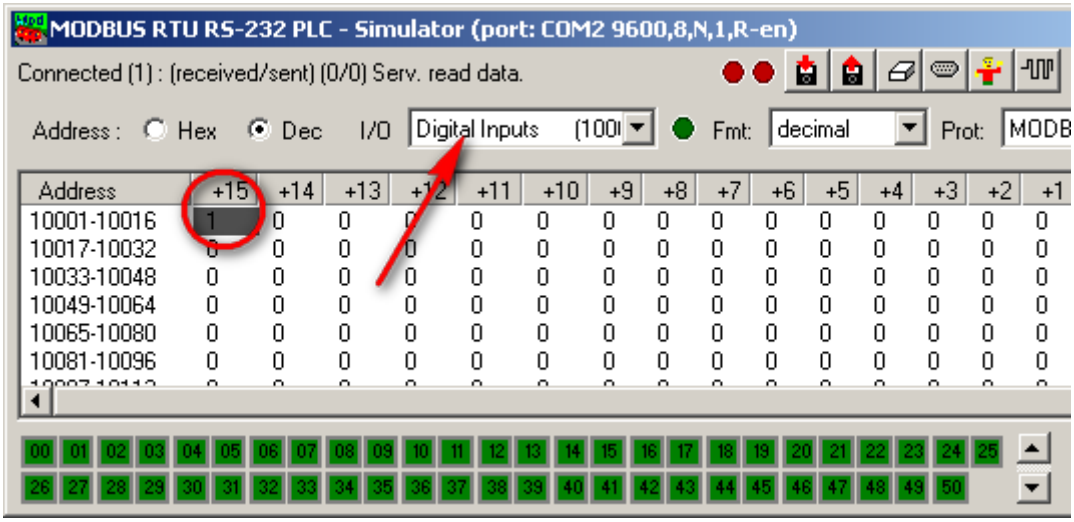


Рис. 64. Відповідність регістрів та сегментів ОА з маппінгом в CoDeSys

Крок 3. Програміст імітує вручну зміну технологічних параметрів (значення на давачах). Для цього потрібно: перейти до сегменту «**Holding Registers**» та у необхідному регістрі (табл. 4.2) записати оновлене значення на давачі. Наприклад, для давача рівня у ємності 1 записати значення у перший регістр зберігання за адресою **4**0001 у fmPLCs.

Крок 4. ВК зчитує поточні значення з давачів та корегує керуючі дії, наприклад, вимикає клапан подачі рідини чи теплоносія. Ручні дії зі зміни значень на інших давачах аналогічні.

Налаштувавши змінні для виведення у **CoDeSys.Trace**, можна спостерігати результати роботи АСК й трасування цих змінних (рис. 4.13).

ЗАУВАЖЕННЯ. Дані у комірках таблиць fmPLCs відображаються у форматі двох байт – слова (тип **WORD**). **Два слова – регістр.**

Часова діаграма показує, як змінюються усі змінні АСК у часі. На графіках LE1 та LE2 видні значення рівнів у ємностях з плином часу. Як тільки у змінну **START** через MODBUS-канал передається логічна 1, що означає увімкнення АСК, починають почергово вмикатися усі виконавчі пристрої. Насос перекачує рідину не миттєво, тож на діаграмі можна спостерігати затримку між відповідними рівнями LE1 та LE2.

4.5 Контрольні запитання

1) В чому полягає сутність та необхідність віртуалізації інтерфейсів?

2) Що таке промисловий протокол MODBUS?

3) Перелічіть основні функції протоколу MODBUS;

4) Перелічіть типи даних протоколу MODBUS;

5) Яка функція протоколу MODBUS зчитує логічні комірки?

6) Що таке маппінг змінних та поясніть його на прикладі CoDeSys.

7) Яка функція протоколу MODBUS записує логічні комірки?

8) Хто ініціює обмін даними в протоколі MODBUS?

4.6 Контрольне завдання

1) Додати до вихідного проекту (з ЛР №1) новий об'єкт ***Додаток*** з назвою відповідно до шаблону **PIP_LR04**. Встановити його активним.

2) Розробити програму АСК за варіантом із ЛР №1, додавши MODBUS-канал у відповідності з методикою та прикладом.

3) Виконати імітацію роботи ОА та емуляцію роботи АСК.

Робочою моделлю ОА може слугувати скрипт з ЛР №3 або ручна імітація за описаним вище алгоритмом.

Результати ЛР подати у вигляді графіків вхідних та вихідних сигналів, побудованих в **CoDeSys.Trace**.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

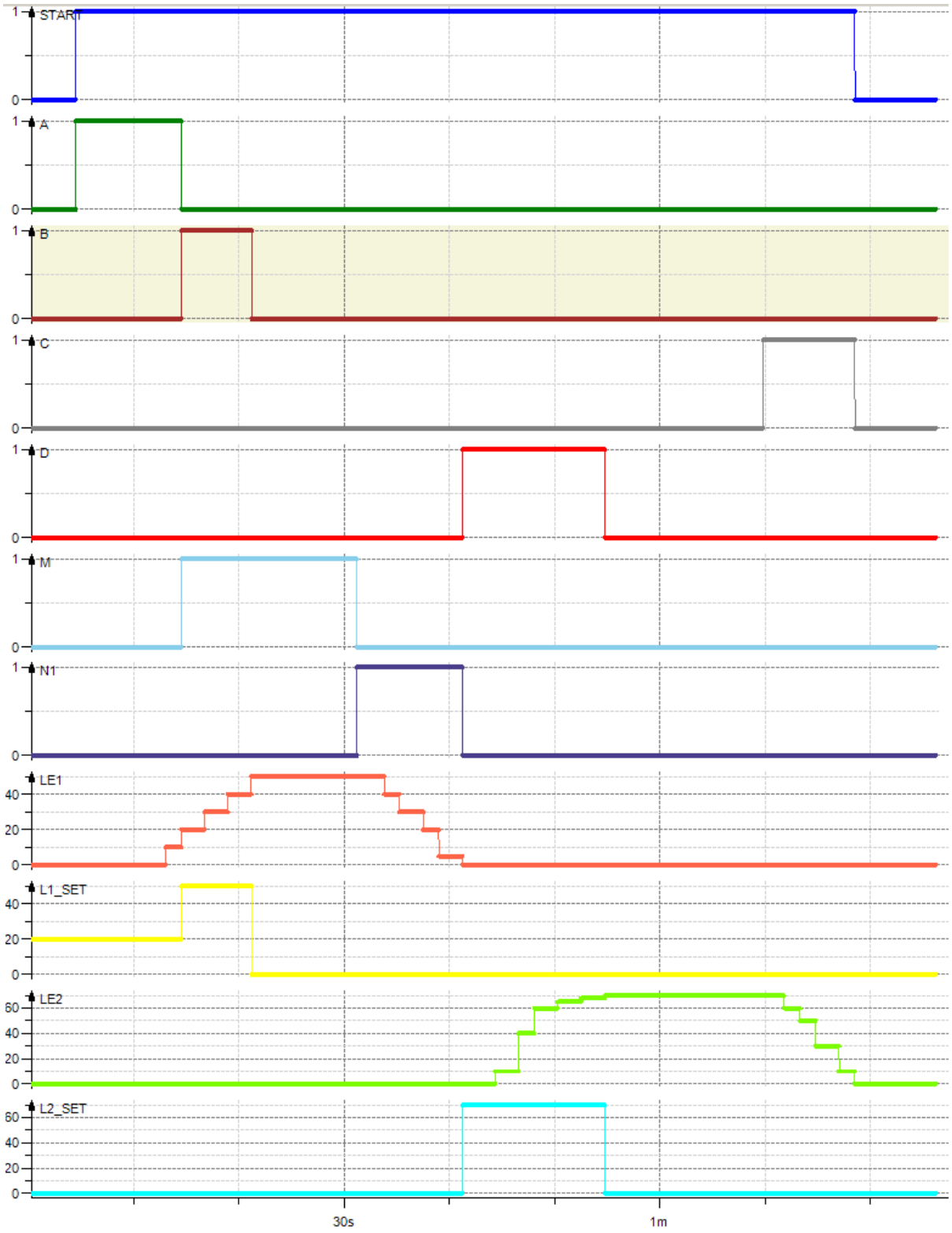


Рис. 4.13. Часова діаграма роботи АСК

5 ЛАБОРАТОРНА РОБОТА № 5

Тема: Розробка локального людино-машинного інтерфейсу HMI

Мета: Вивчити вбудовані засоби візуалізації в середовищі CoDeSys для створення Екранів HMI локального місця оператора (ЛМО).

Завдання:

1) Засвоїти методику розробки статичної та динамічної візуалізації в CoDeSys, а саме створення графічних примітивів HMI-інтерфейсу для ВК;

2) Виконати контрольне завдання до ЛР №5.

5.1 Постановка задачі розробки людино-машинного інтерфейсу

Розробити примітив HMI-екрану з базовими індикаторами.

1) Використати за базу робочі вихідні проекти з ЛР №3-4;

2) Розробити статичне зображення устаткування АСК;

3) Додати та налаштувати динамічні зображення та елементи;

4) Виконати імітацію роботи АСК із ВОА.

Структура стенду наведена на рис. 5.1.

Лабораторний стенд на базі ПК складається із наступних елементів:

1) Середовище програмування CoDeSys;

2) ВК – **CoDeSys Control Win**.

3) Віртуальний RS-232 – «**ELTIMA Virtual COM**» – VCOM;

4) Засоби візуалізації HMI у середовищі CoDeSys;

5) ВОА – **ручний (або скриптовий) імітатор** ОА в fmPLCs;

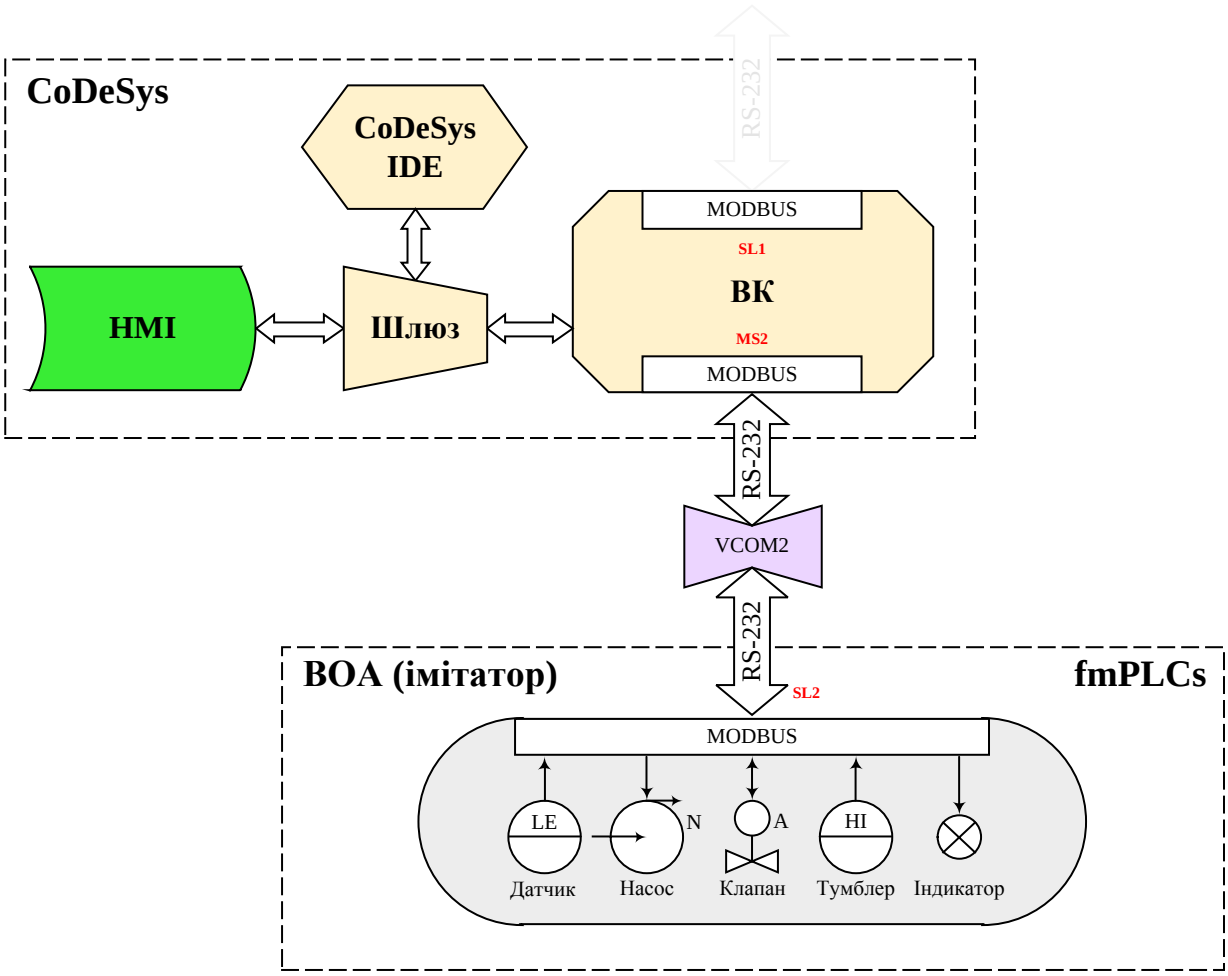


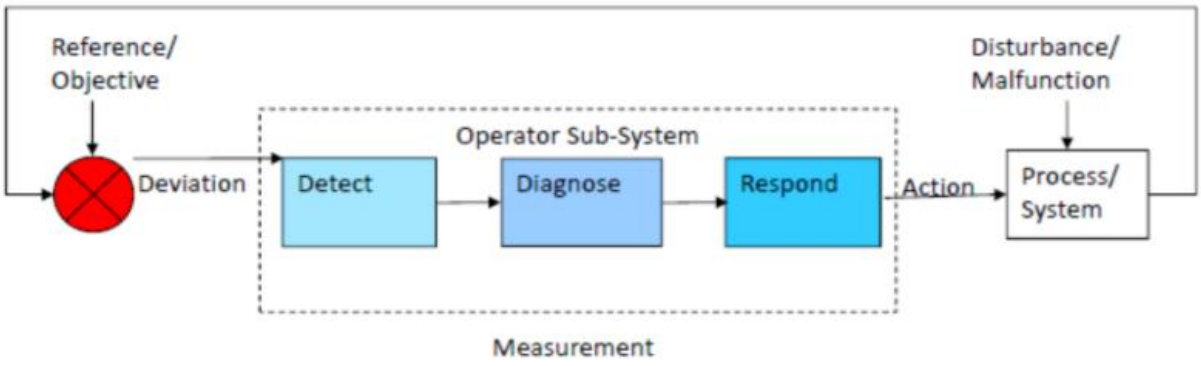
Рис. 5.1. Структура віртуального лабораторного стенду №5

5.2 Теоретичні відомості про HMI

HMI – це набір всіх технічних засобів, які дозволяють людині втручатися в поведінку АСК. Як правило, HMI – це комп'ютер з графічним дисплеєм, який візуально відображає поведінку системи, і користувач має можливість втручатися в АСК. Тим не менш, HMI може бути найпростішим пультом дистанційного керування набором тумблерів і світлодіодних індикаторів.

Існують загальні принципи впровадження HMI рішень. HMI є ефективним інструментом для безпечного і ефективного управління процесом. HMI допомагає в ранньому виявленні, діагностиці та правильному реагуванню на нештатні ситуації. HMI побудована так, щоб допомогти операторам визначити пріоритети реагування на основні абодекілька одночасних

системних збоїв. Відмова відеокадра (дисплея) або елементів на ньому відразу очевидні для оператора.



HMI повинні бути засновані на вимогах завдань і потреб оператора. Так як існують різні користувачі HMI, кожну з їхніх потреб слід враховувати при проектуванні. Важливо, щоб потреби первинних користувачів (операторів заводу) мали пріоритет над інтересами всіх інших користувачів.

Основна вимога до HMI-пристроїв: оператор повинен «визначити ->діагностувати –> реагувати». Потім оцінити результати дії. Оператор має усвідомолювати ситуацію що відображає HMI: бути в курсі того, що відбувається в цьому процесі, розуміння стану процесу в даний час, розуміння ймовірного стану процесу в майбутньому. Коли процес працює як очікувалося, дисплей повинен мати мінімум сенсорних стимулів. Коли процес відхиляється від очікувань, тоді HMI повинен забезпечувати візуальні і/або звукові сигнали з відповідною інформативністю для ситуації.

Основні принципи взаємодії із користувачем (оператором): послідовність у виконанні на всіх режимах взаємодії; своєчасний зворотній зв'язок для введення даних і управління діями; добре налагоджена взаємодія з користувачем (мінімізовано кількість виборів або кількість натискань клавіш); використання відповідних особливостей для повідомлень про помилки.

5.3 Приклад виконання роботи

Далі додати візуалізацію в проект CoDeSys (локалізація **Russian**):

«Application → Добавить объект → Визуализация».

Вказавши ім'я і натиснувши «**Добавить»**, буде додано візуалізацію. Відкривши візуалізацію подвійним натисненням ЛКМ, відкриється вкладка робочого поля, а справа з'явиться «**Панель инструментов»** (рис. 5.2), що містить всі необхідні елементи для здійснення візуалізації.

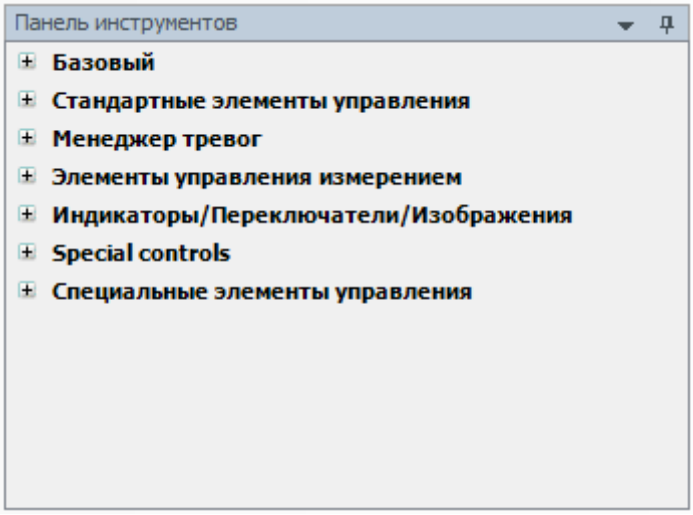


Рис. 5.2. Панель інструментів для візуалізації

Розробка статичного зображення Екрану HMI

Для створення ліній використовується відповідний «**Базовый»** елемент «**Линия»** . В «**Свойства»** лінії (рис. 5.3) треба задати товщину – для товстих ліній 2, а тонких 1.

Тип элемента	Линия
Позиция	
Цвета	
Вид элемента	
Ширина линии	2
Стиль линии	Сплошной

Рис. 5.3. Товщина ліній

Для зображення клапанів використовується елемент «**Полигон»** .

Для давачів, індикаторів та насосу використовується елемент «**Эллипс»** з відповідними радіусами: для індикаторів 25, для давачів 50, для насосу 40.

Для стрілок використовується елемент «**Полигон»** , проте зі встановленим кольором заливки (рис. 5.4).

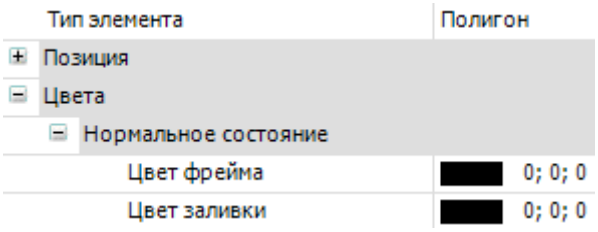


Рис. 5.4. Заливка полігону

Для створення ємностей можна скористатися елементом «**Прямоугольник**» , проте для ємностей треба з'єднати 4 лінії, розташовані у вигляді прямокутника, потім виділити їх, а далі в контекстному меню згрупувати їх в 1 елемент (рис. 5.5).

Це зроблено для того, щоб фігура була «прозорою», тобто не закривала інші елементи, розташовані під нею. Це можна також зробити за допомогою елементу «**Ломаная**» .

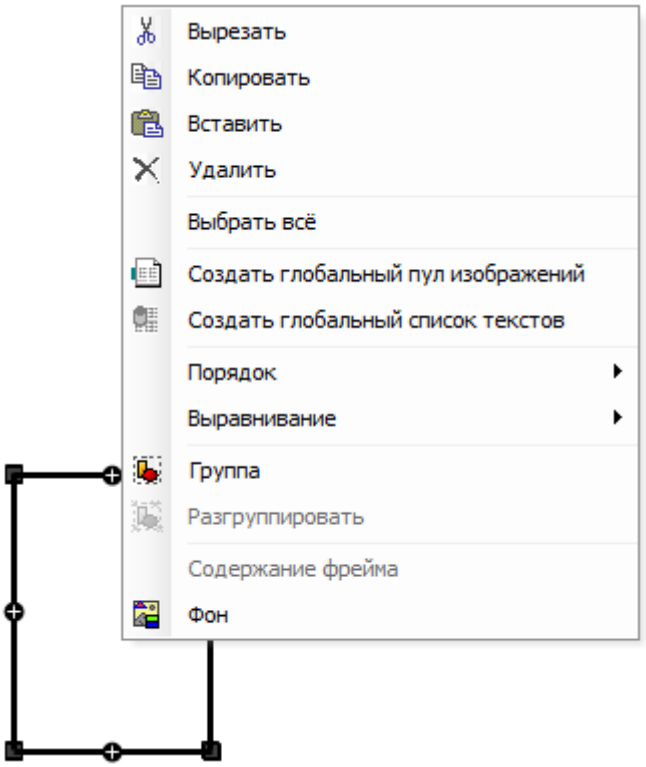


Рис. 5.5. Групування елементів

Для створення підписів використовується елемент «**Панель инструментов → Стандартные элементы управления → Метка**» . Треба обрати розмір та тип шрифту (рис. 5.6).

Тип элемента	Метка
⊞ Тексты	
⊞ Позиция	
⊟ Свойства текста	
Горизонтальное выравнивание	По центру
Вертикальное выравнивание	По центру
Шрифт	Arial; 14,25pt
Цвет шрифта	0; 0; 0

Рис. 5.6. Налаштування властивостей тексту

Аналогічно налаштовується текст і для інших елементів Екрану.

Приклад статичного Екрану HMI, що відповідає рис. 2.1 виглядатиме так як зображено на наступному рис. 5.7 (давач LS2 відсутній).

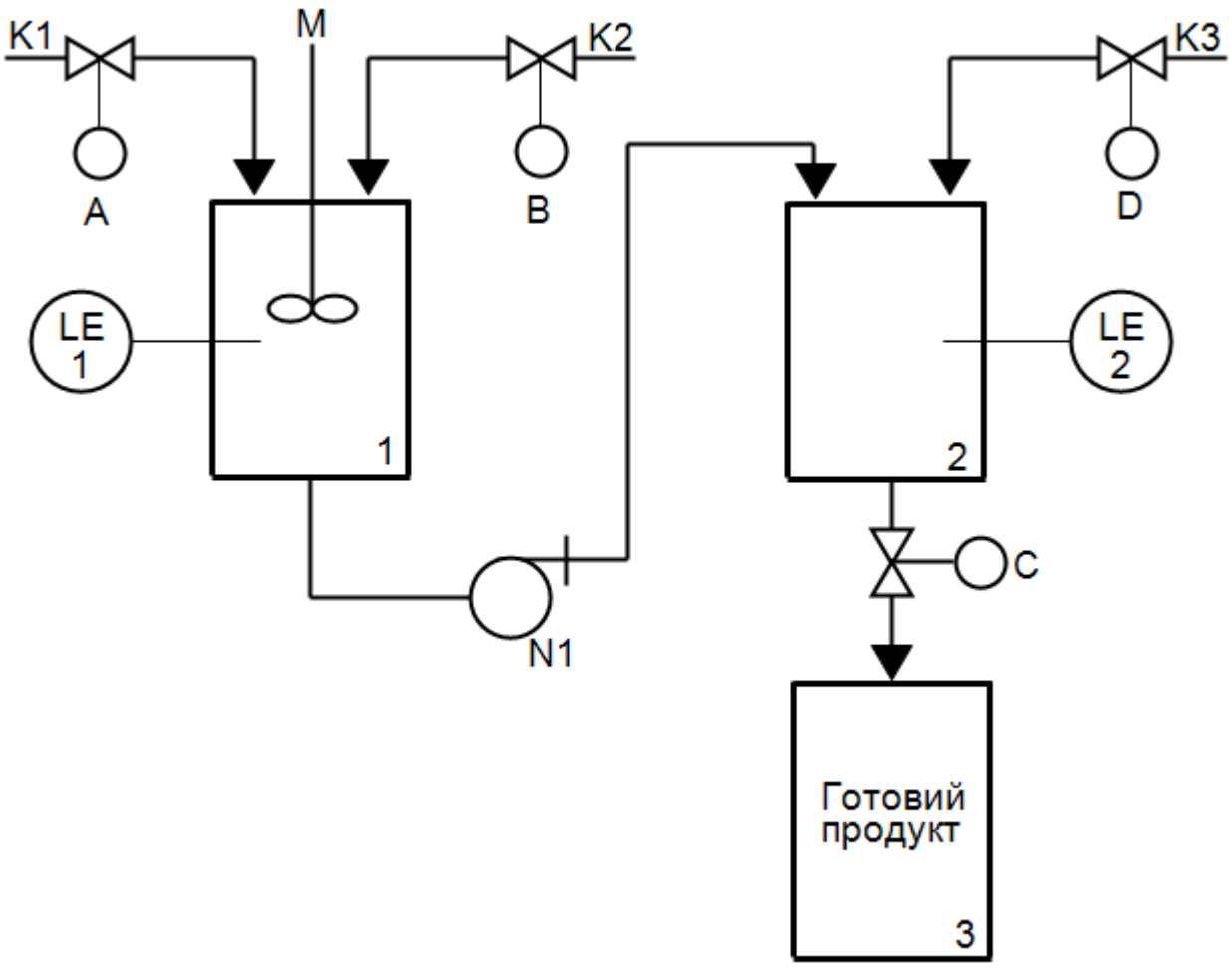


Рис. 5.7. Статична візуалізація Екрану HMI АСК

Додавання динамічних елементів до Екрану HMI

Поверх створених раніше індикаторів, що зображуються еліпсами, треба додати елементи типу **«Панель инструментов → Индикаторы/ Переключатели/ Изображения → Индикатор»**. У властивостях елементу треба прив'язати відповідну змінну (рис. 5.8) зі словника змінних АСК (табл. 4.1). Цю таблицю потрібно використовувати і надалі для прив'язування змінних.

Рис. 5.8. Прив'язка змінної до індикатору

У властивостях клапанів (рис. 5.9) необхідно вказати змінну типу BOOL, яка відповідає за динамізацію, а також задати кольори двох станів – нормальний (змінна = FALSE) та стан тривоги (змінна = TRUE).

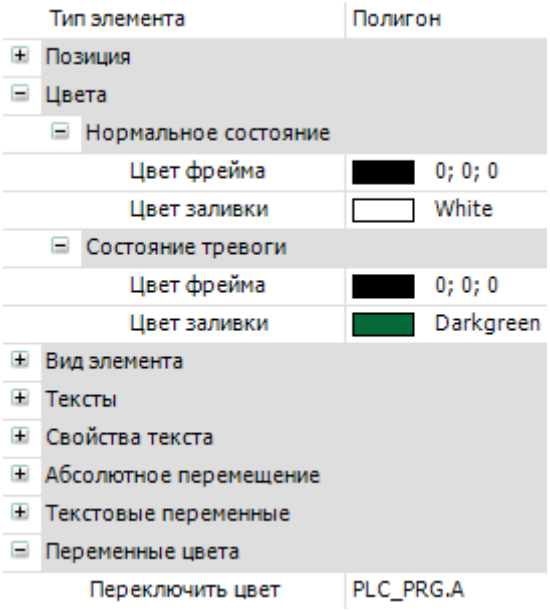


Рис. 5.9. Налаштування динамізації клапанів

Аналогічно налаштовується графічна динамізація мішалки (рис. 5.10).

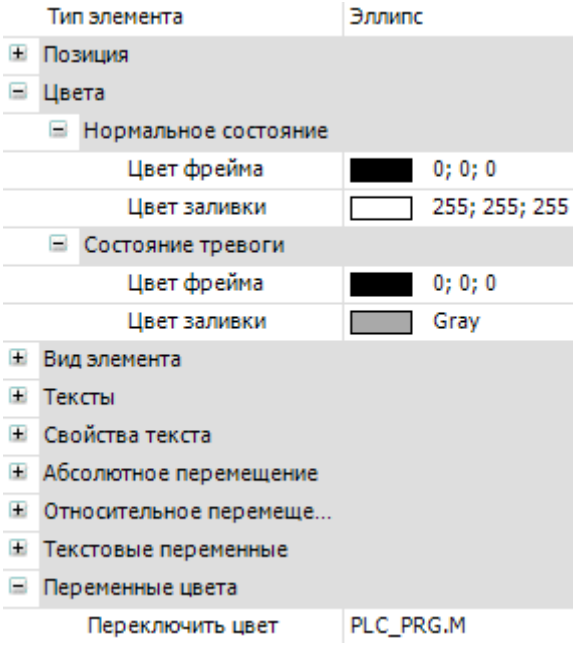


Рис. 5.10. Налаштування динамізації мішалки

Для динамізації рівня в ємностях треба скористатися елементами типу **«Панель инструментов → Элементы управления измерением → Гистограмма»**  . Необхідно розташувати гістограми під раніше розміщеними статичними ємностями, щоб контури прямокутника накладалися на контури гістограми. Керування шарами розміщення об'єктів здійснюється аналогічно функціоналу **Microsoft Visio** (рис. 5.11).

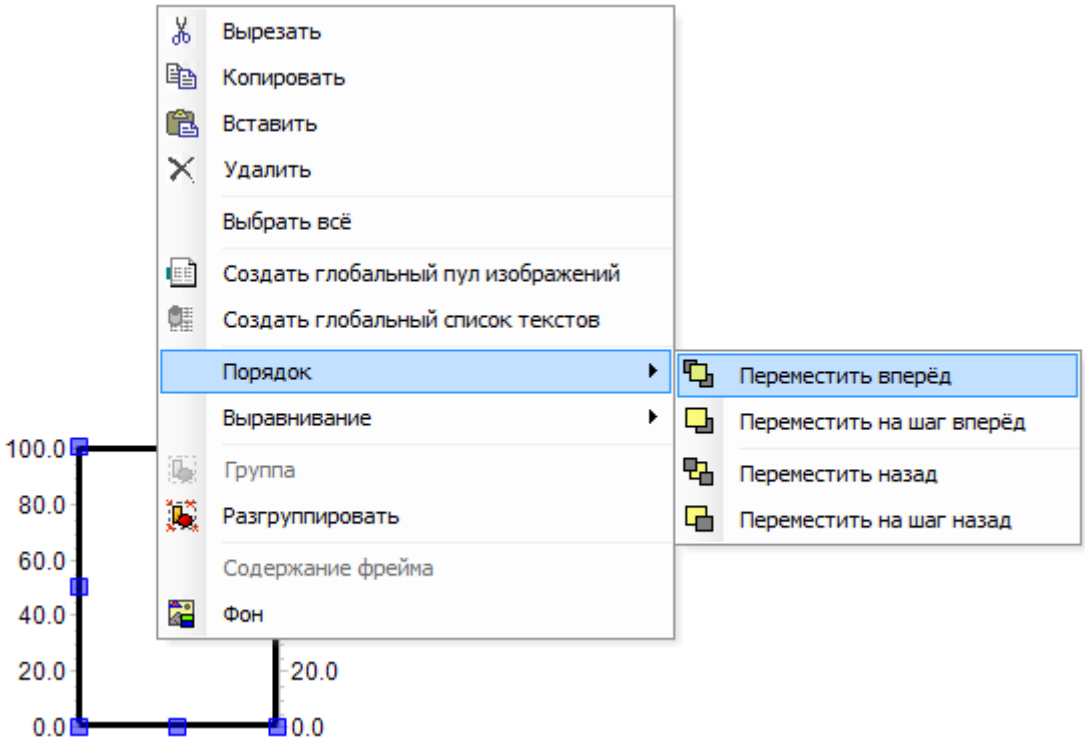


Рис. 5.11. Керування шарами розміщення об'єктів

В налаштуваннях властивостей гістограми треба вказати змінну, що відповідає за рівень рідини в конкретній ємності, а також встановити ширину 100 (рис. 5.12).

Тип элемента	Гистограмма
Массив данных	PLC_PRG.LE1
Тип отображения	Линейки
Показывать горизонтальные линии	☐
Относительная ширина линейки	100

Рис. 5.12. Налаштування ємностей

Щоб відобразити поточне значення **LE1** та **LE2** використовуються елементи «**Прямоугольник**» . В полі «**Текст**» вказуються директиви форматування даних (рис. 5.13), що відповідають функції **`sprintf()`** зі стандартної *C* бібліотеки (`%d` – відображення десяткових чисел):

Тип элемента	Прямоугольник
⊞ Позиция	
⊞ Цвета	
⊞ Вид элемента	
⊟ Тексты	
Текст	%d
Подсказка	
⊞ Свойства текста	
⊞ Абсолютное перемещение	
⊞ Относительное перемещение	
⊟ Текстовые переменные	
Текстовая переменная	PLC_PRG.LE1

Рис. 5.13. Налаштування відображення десяткових значень

Для відображення стану вимикача системи використовується елемент **«Панель инструментов → Индикаторы/ Переключатели/ Изображения → Переключатель»** (рис. 5.14).

Тип элемента	Переключатель
⊞ Позиция	
Переменная	PLC_PRG.START
Поведение элемента	Переключатель изображения

Рис. 5.14. Налаштування перемикача

Для задавання значень levelA, levelB та levelC використовується елемент **«Панель инструментов → Стандартные элементы управления → Комбинированное окно - Целочисленный»** . Потрібно вказати інтервал можливих значень (рис. 5.15).

Тип элемента	Комбинированное окно - Целочисленный
⊞ Позиция	
Переменная	PLC_PRG.levelA
Список текстов	
Пул изображений	'1-100'
Максимальное значение	100

Рис. 5.15. Налаштування уставок ComboBox

Приклад динамічного Екрану HMI, що відповідає рис.2.1 виглядатиме наступним так як зображено на рис. 5.16.

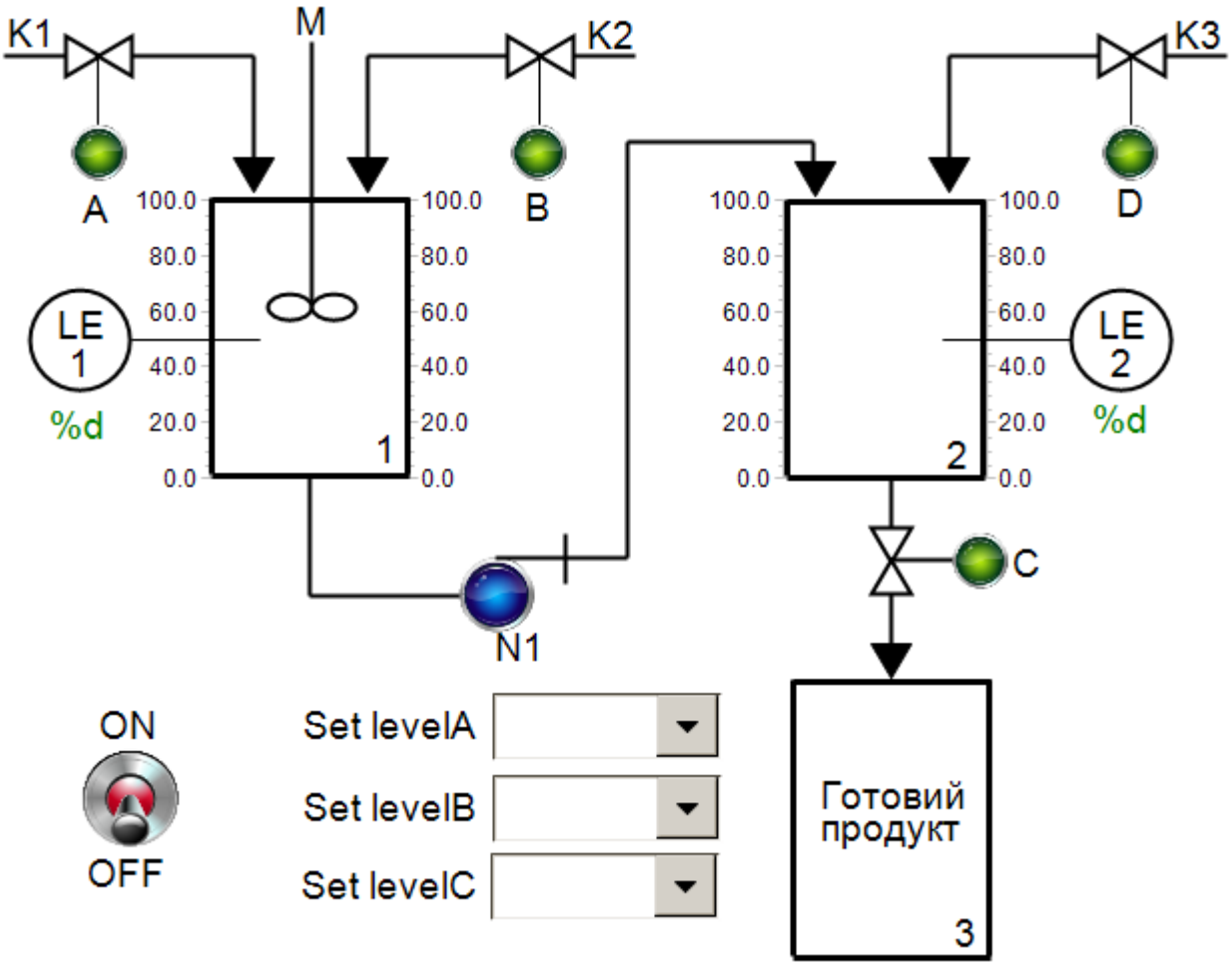


Рис. 5.16. Візуалізація Екрану HMI

Запуск проекту на виконання

Запускається проект за допомогою **«Online → Login»**, а далі **«Debug → Start»**. За замовчуванням у комбінованих вікнах (ComboBox) встановлено значення компонентів *К1*, *К2* та *К3*, що відповідають заданим в програмі значенням (уставки). Проте користувач може змінити ці значення скориставшись відповідними комбінованими вікнами.

Запуск АСК відбувається на рівні ВОА. В програмі **fmPLCs** до біту Coils, що відповідає за тумблер «**СТАРТ**» (табл. 2.1) вводиться одиниця. Відразу після запуску АСК, на Екрані HMI стан перемикача змінюється на **ON** та відкривається клапан **A**, про що свідчить підсвітка клапану та індикатору-

коло (привод клапану). Після відкриття клапану **A** очікується заповнення ємності *1* компонентою *K1*, що здійснюється ручною імітацією – введенням значень у відповідний регістр **fmPLCs.** Ввівши значення **LE1** (наприклад, 18) можна спостерігати стан системи як на рис. 5.17.

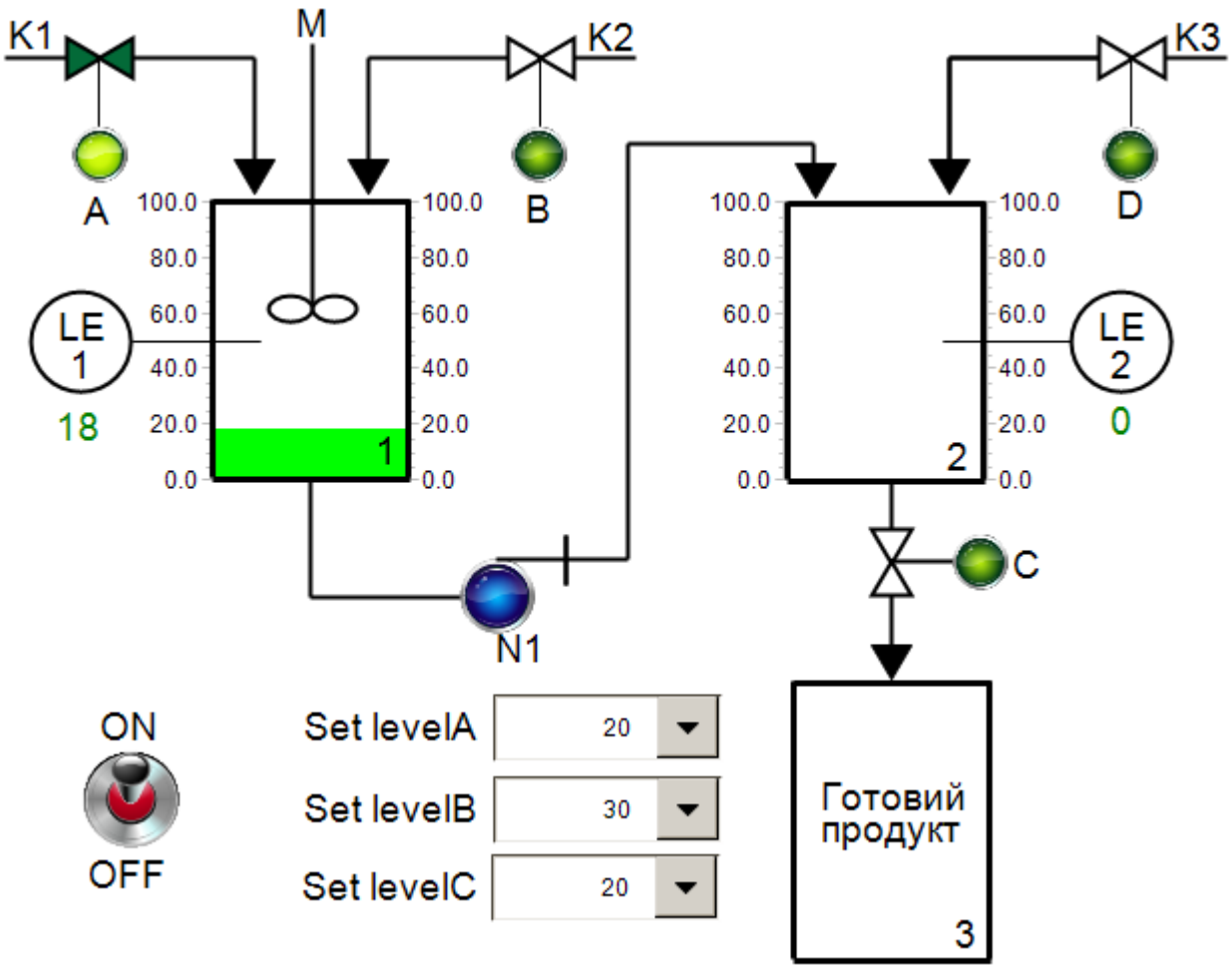


Рис. 5.17. Візуалізація завантаження компоненту *K1*

Відповідно до алгоритму, що описаний у постановці задачі до ЛР №2, можна спостерігати за: зміною станів решти клапанів, індикаторів параметрів, наповненням ємностей та роботою насосу. Якщо віртуальний тумблер «**СТАРТ**» на рівні ВОА своєчасно не вимкнути, тоді відбудеться повторний перезапуск циклу роботи АСК.

5.4 Контрольні запитання

1) В чому полягає відмінність між мовами CFC та FBD?

2) Яке призначення ФБ «таймер»?

3) За якої умови вмикається таймер?

4) Як за допомогою мови IL перевірити, чи ввімкнений таймер?

5) Як в пакеті CoDeSys створити візуалізацію?

6) За допомогою якої панелі користувач може вибирати геометричні фігури для створення візуалізації системи?

7) Як виконати зв'язок між значенням змінної та кольором фігури?

5.5 Контрольне завдання

1) Додати до вихідного проекту (з ЛР №1-4) новий об'єкт ***Візуалізація*** з назвою відповідно до шаблону **PIP_LR05**. Встановити вихідний ***Додаток*** **PIP_LR05** активним.

2) Розробити HMI, себто графічний Екран АСК, використовуючи вбудовані засоби візуалізації CoDeSys.

3) Відобразити на HMI значення технологічних параметрів, стани керуючих сигналів та всіх давачів. А також налаштунки програмних задавачів і параметри контролю параметрів (за наявності у завданні).

4) Врахувати залежність форми ємностей ОА, діапазон зміни керуючих сигналів та погрішність давачів (за наявності у завданні).

5) Виконати імітацію роботи АСК.

Результати ЛР подати у вигляді скріншотів динаміки роботи HMI.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

6 ЛАБОРАТОРНА РОБОТА № 6

Тема: Об'єктно-орієнтована програма керування

Мета: Розробка керуючої програми для віртуального контролера CoDeSys засобами об'єктно-орієнтованого програмування (ООП).

Завдання:

1) Виконати розробку керуючої програми контролера АСК.

2) Виконати контрольне завдання до ЛР №6.

6.1 Постановка задачі розробки об'єктної програми керування

1) Написати інтерфейси та ФБ що їх реалізують для різних частин системи: клапанів, датчиків та приводів (наприклад IValve, ISensor, IMotor та ін.);

2) У керуючій програмі замінити змінні що описують датчики, клапани та приводи викликами методів написаних ФБ;

3) Налаштувати новий маппінг значень комірок MODBUS на внутрішні змінні написаних ФБ.

4) Продемонструвати роботу отриманої керуючої програми АСК.

6.2 Теоретичні відомості

Методологія розробки керуючих програм до ВК базується на знанням, уміннях і досвіду, що отримані студентом при вивченні дисциплін ОП, ООП, МПС.

6.3 Приклад виконання роботи

Перед початком роботи необхідно включити підтримку ООП в CoDeSys: «**Tools → Options… → Features**».

Після виконання наведених дій у меню «**Add Object**» об'єкту «**Application**» з'явиться пункт «**Interface...**» що надасть можливість створювати інтерфейси.

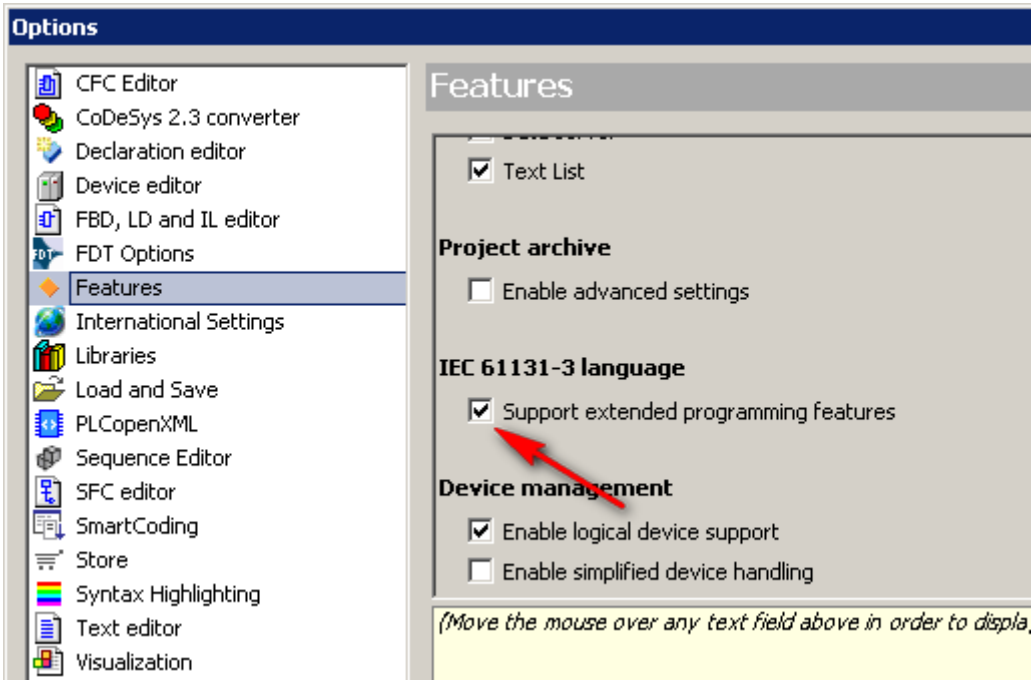


Рис. 1. Дозвіл на підтримку стандарту IEE 61131-3 у CoDeSys

Для прикладу, створимо інтерфейс **IValve** що описує простий клапан що має лише 2 стани – відкритий та закритий. Натискаємо ЛКМ «**Application → Add Object → ПКМ → Interface...**».

До створеного інтерфейсу додамо 2 методи – *open* та *close*, що не приймають параметрів та нічого не повертають. Натискаємо ЛКМ «**IValve → Add Object → ПКМ → Interface method...**».

Далі необхідно створити ФБ, що реалізує цей інтерфейс.

```
FUNCTION_BLOCK Valve IMPLEMENTS IValve
VAR_INPUT    // Вхідні змінні
END_VAR
VAR_OUTPUT   // Вихідні змінні
END_VAR
VAR    // Внутрішні змінні, доступні з коду ВК та методів
     isClosed: BOOL := FALSE;
END_VAR
```

Розділ реалізації ФБ `Valve`:

```
METHOD open
isClosed := FALSE;

METHOD close
```

```
isClosed := TRUE;
```

У результаті, отриманий ФБ можна підключити у керуючу програму та керувати клапанами АСК за допомогою викликів *open*() та *close*(). Аналогічно, для виконавчих елементів АСК можна реалізувати ФБ *Motor* що має 2 методи – *start*() та *stop*().

Розробимо ФБ для простого булевого датчика: створимо інтерфейс *ISensor* що має єдиний метод – *getValue*() , що повертає тип BOOL.

```
FUNCTION_BLOCK Sensor IMPLEMENTS ISensor
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
     value: BOOL := FALSE; // Значення надходить з ОА
END_VAR
```

Розділ реалізації ФБ `Sensor`:

```
METHOD getValue : BOOL
getValue := value;
```

Розробимо просту керуючу програму з використанням написаних ФБ на прикладі алгоритму роботи ОА з ЛР в МПС та ПрІС-1.

В розділі змінних PLC_PRG помістимо наступний фрагмент коду:

```
PROGRAM PLC_PRG
VAR
     startBtn: Button;
     valveA, valveB, valveC, valveD: Valve;
     motorM, pump: Motor;
     sensorLS1, sensorLS2: Sensor;
     timer1, timer2, timer3, timer4, timer5: TON;
END_VAR
```

Та у розділі реалізації наберемо наступну схему:

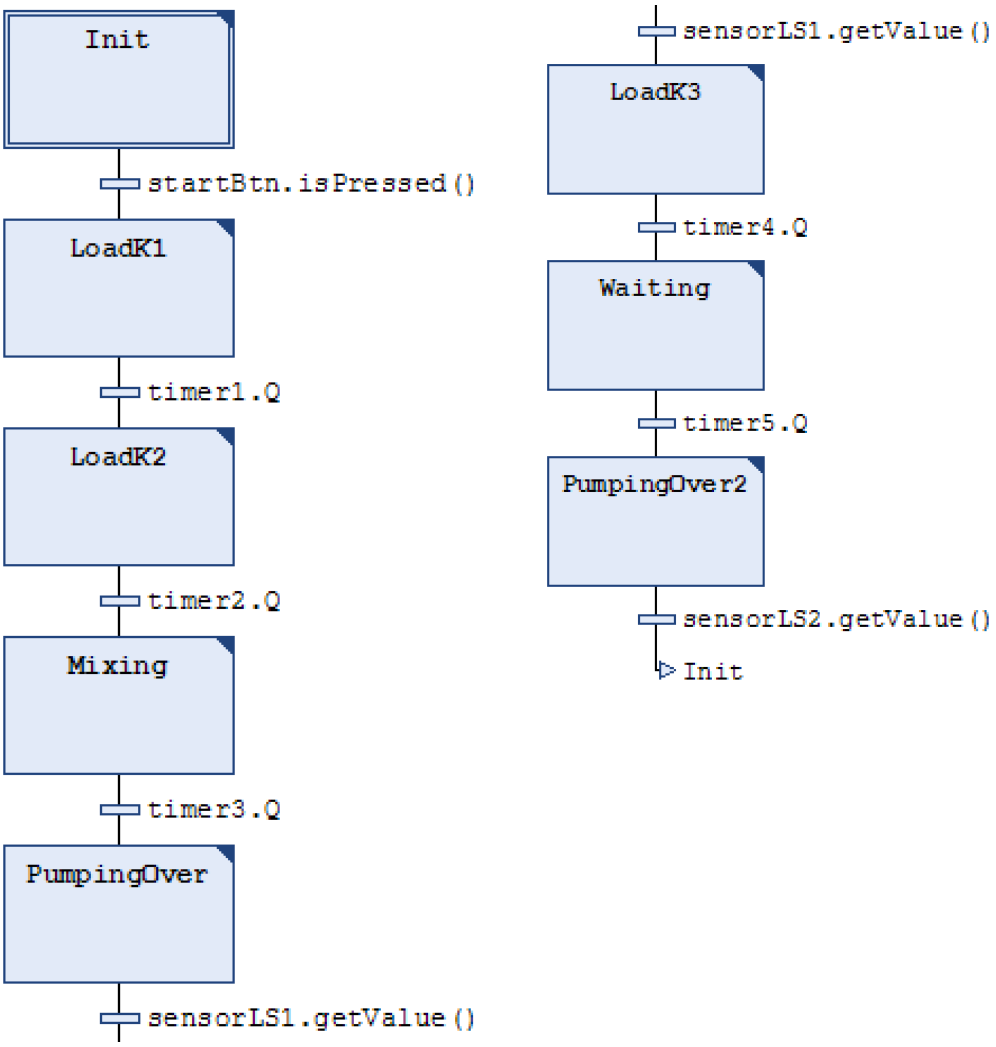


Рис. 2. Схема SFC програми керування АСК

Реалізація кожного кроку наступна:

Крок `Init`

```
valveA.close();
valveB.close();
valveC.close();
valveD.close();
motorM.stop();
pump.stop();
```

Крок `LoadK1`

```
valveA.open();
motorM.start();
timer1(IN := TRUE, PT := t#10s);
```

Крок `LoadK2`

```
valveA.close();
```

```
valveB.open();
timer2(IN := TRUE, PT := t#15s);
```

Крок `Mixing`

```
valveB.close();
timer3(IN := TRUE, PT := t#10s);
```

Крок `PumpingOver`

```
motorM.stop();
pump.start();
```

Крок `LoadK3`

```
pump.stop();
valveD.open();
timer4(IN := TRUE, PT := t#10s);
```

Крок `Waiting`

```
valveD.close();
timer5(IN := TRUE, PT := t#15s);

// PumpingOver2
valveC.open();
```

Для зручності, інтерфейси та ФБ що їх реалізують можна зберігати в окремих директоріях проекту – **Interfaces** та **Components**.

6.4 Контрольні запитання

1) У чому перевага використання засобів ООП у CoDeSys?
2) Яке призначення інтерфейсів?
3) Які процеси та об'єкти можна представити ООП-моделлю?

6.5 Контрольне завдання

Розробити (модифікувати) програму керування ОА із використанням ООП на базі вихідного індивідуального завдання, наведеного у ЛР №1.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

7 ЛАБОРАТОРНА РОБОТА № 7

Тема: Проектування регулятора та програмного задавача

Мета: Синтез та програмна реалізація регулятору.

Завдання:

1) Виконати синтез та розробку програмного регулятора АСК.

2) Виконати контрольне завдання до ЛР №7.

7.1 Постановка задачі проектування регулятора

1) Виконати аналіз контуру регулювання ОА.

2) На базі аналітичних результатів обрати структуру регулятора.

3) Синтезувати та розрахувати модель регулятора.

4) Реалізувати програмно регулятор у ВК засобами CoDeSys.

5) Перевірити роботу регулятора на контролерному рівні з ВОА.

7.2 Теоретичні відомості про регулятори

Методологія розробки програмних регуляторів базується на знанням, уміннях і досвіду, що отримані при вивченні дисциплін КСУ та ТШІУ.

У якості регулятора розглянемо двопозиційний релейний регулятор (ДРР), статична характеристика якого із зоною невизначеності має вигляд як показано на рис. 6.1.

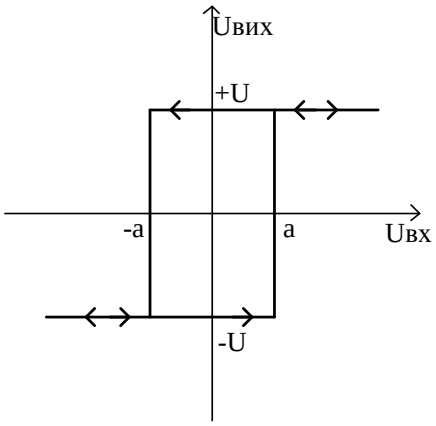


Рис. 6.1. Перехідна характеристика двопозиційного релейного регулятора

Стан елементу поза інтервалом [-a; a] можна чітко визначити. На інтервалі [-a; a] вихід елемента визначається попереднім станом релейного елементу, тобто за допомогою похідної за часом від вхідного сигналу.

Таким чином, поведінку ДРР можна записати у вигляді правил:

- Якщо **Uвх> a** то **Uвих = +U**;
- Якщо **Uвх <-a** то **Uвих = -U**;
- Якщо **-a <Uвх <a** і **dUвх / dt> 0** то **Uвих = -U**;
- Якщо **-a <Uвх <a** і **dUвх / dt <0** то **Uвих = +U**.

Дискретні величини приймемо наступні: +U=1, а –U=0.

Також, з рис. 1 бачимо, що рівень компоненту в ємності *1* можна регулювати вимикаючи клапан **A** та включаючи насос **N1** (зменшення рівня компоненту в ємності) і навпаки (збільшення компоненту в ємності).

7.3 Приклад виконання роботи

ЛР базується на результатах, отриманих протягом виконання ЛР №3. Для впровадження регулятора в логіку роботи ВК необхідно виконати три послідовних кроки:

- Обрати регульовану величину;
- Визначитися з типом регулятора;
- Обрати регулюючі інструменти.

На даному прикладі (схема АСК), регульованою величиною виступає рівень компоненту в ємності *1*. Оберемо бажане значення рівня компоненту в ємності *1* за 5.5, а межі коливання похибки за 1 – це значить, що значення рівня компоненту з використанням регулятора має коливатися в межах від 4.5 до 6.5.

Так як необхідно реалізувати регулювання обраної регульованої величини засобами середовища CoDeSys, створимо ФБ, обравши мову ST. Розглянемо змінні, необхідні для реалізації подібного ФБ (табл. 1).

Таблиця 1

Змінна	Тип	Опис	Область
regulationNeeded	BOOL	Значення, яке приймає ДРР в ході виконання функції. На основі цього значення приймається рішення чи потрібно регулювати рівень компоненти в ємності *1*, чи ні.	Локальна змінна
last_value	BOOL	Останнє значення, яке приймала змінна regulationNeeded	Глобальна зміна
last_e	REAL	Останнє значення, яке приймала помилка (змінна e)	Глобальна змінна
level	REAL	Поточне значення рівня компоненту	Вхідна змінна
opt_level	REAL	Бажане значення рівня компоненту	Вхідна змінна
range	REAL	Межі коливання помилки	Вхідна змінна
valve_A state_A	REAL	Стан клапану **A**	Вихідна змінна
Pump state_N1	REAL	Стан насосу **N1**	Вихідна змінна
prev_e	REAL	Допоміжна змінна для зберігання попереднього значення помилки	Локальна змінна
e	REAL	Допоміжна змінна для зберігання поточного значення помилки	Локальна змінна

Реалізація функції REGULATOR:

```
VAR_GLOBAL
      last_value:BOOL;
      last_e:REAL;
END_VAR

FUNCTION_BLOCK REGULATOR
//Вхідні змінні
VAR_INPUT
      level:REAL;
      opt_level:REAL;
      range:REAL;
END_VAR
//Вихідні змінні
VAR_OUTPUT
      state_A:REAL;
      state_N1:REAL;
END_VAR
//Внутрішні змінні
```

```
VAR
      regulationNeeded:BOOL;
      prev_e:REAL;
      e:REAL;
END_VAR

//Реалізація регулятора
e:=opt_level-level;
prev_e:=last_e;
last_e:=e;
IF e=0 THEN
      RETURN;
ELSE
      IF e>range THEN
            regulationNeeded:=TRUE;
            last_value:=regulationNeeded;
      ELSE
            IF e<(-1)*range THEN
                  regulationNeeded:=FALSE;
                  last_value:=regulationNeeded;
            ELSE
                  IF e>0 AND prev_e>0 THEN
                        IF e>=prev_e THEN
                              regulationNeeded:=TRUE;
                              last_value:=regulationNeeded;
                        ELSE
                              regulationNeeded:=last_value;
                        END_IF
                  ELSE
                        IF e<0 AND prev_e<0 THEN
                              IF e<=prev_e THEN
                                    regulationNeeded:=FALSE;
                                    last_value:=regulationNeeded;
                              ELSE
                                    regulationNeeded:=last_value;
                              END_IF
                        ELSE
                              regulationNeeded:= NOT last_value;
                              last_value:=regulationNeeded;
                        END_IF
                  END_IF
            END_IF
      END_IF
END_IF
IF regulationNeeded=FALSE THEN
      state _A:=0;
      state_N1:=5;
ELSE
      valve_A:=0.2;
      state_N1:=0;
END_IF
```

Наступним кроком буде включення виклику написаного ФБ в одну з SFC дій.

- Оголошення нової змінної в блоці **PLC_PRG** [Device: PLC Logic: Application] (рис. 6.2).

```
    levelB:REAL:=6.2;
    levelC:REAL:=8.5;
    temp:TIME;
    real2words:REAL_TO_WORDS;
    reg:REGULATOR;
END_VAR
VAR_OUTPUT
    t:TIME;
END_VAR
```

Рис. 6.2. Декларування змінної «Регулятор»

- Додання виклику ФБ в логіку роботи програми ВК (рис. 6.3).

PLC_PRG.GS2_active

```
1  A:=1;
2  temp:=GS2.t;
3  LE1:=WORDS_TO_REAL(LE1_h,LE1_l);
4  LE2:=WORDS_TO_REAL(LE2_h,LE2_l);
5  real2words(real_var:=A,hw=>A_h,lw=>A_l);
6  reg(level:=le1,opt_level:=5.5,range:=1,valve_A=>A,pump=>N1);
7  Real2words(real_var:=A,hw=>A_h,lw=>A_l);
8  real2words(real_var:=N1,hw=>N1_h,lw=>N1_l);
```

Рис. 6.3. Виклик коду регулятора в тілі керуючої програми

Рядок

reg(level:=le1,opt_level:=5.5,range:=1,valve_A=>A,pump=>N1); відповідає за виклик написаного попередньо ФБ.

Рядки

Real2words(real_var:=A,hw=>A_h,lw=>A_l);

та **real2words(real_var:=N1,hw=>N1_h,lw=>N1_l);** відповідають за оновлення отриманих значень через MODBUS-канали в регістрах fmPLCs.

Трейсування змінних дає наступні результати (рис. 6.4).

Як бачимо з рис. 2, спочатку рівень компоненту в ємності збільшується, проте як тільки значення рівня компоненти перевищує задане, включається регулювання та рівень компоненту зменшується. В результаті рівень компоненту коливається в межах дозволеного проміжку.

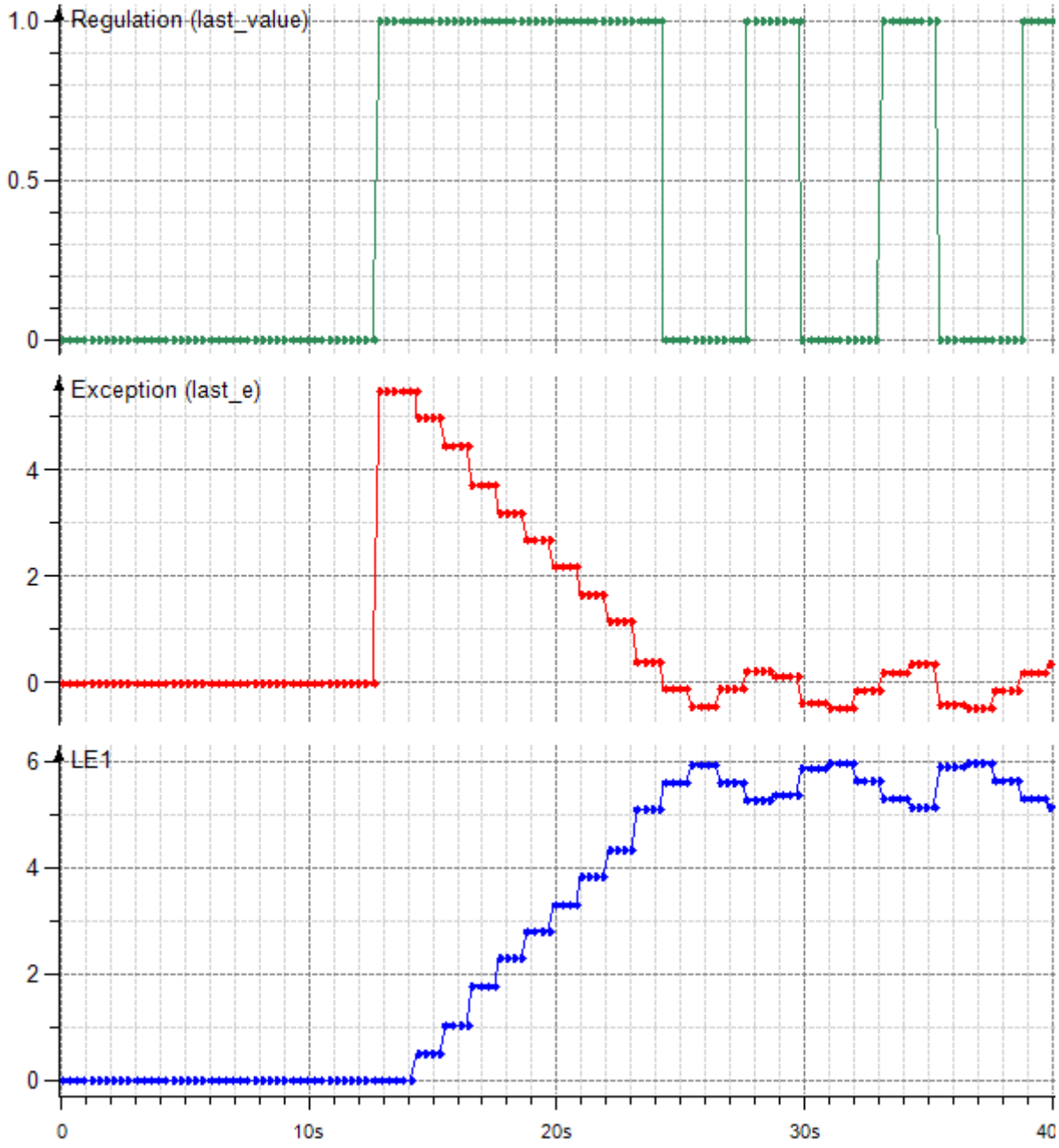


Рис.6.4. Відпрацювання регулятора в АСК

7.4 Контрольні питання

1) Що таке двопозиційний релейний регулятор?

2) За яким принципом обирається контур регулювання та регулятор?

3) Які кроки є необхідними для впровадження та реалізації регулятора в АСК?

4) Яким чином відбувається регулювання рівня компоненту в ємності *1*?

5) Як впровадження регулятора впливає на реалізацію ОА, що виконується в середовищі fmPLCs?

7.5 Контрольне завдання

Синтезувати регулятор для ВК на базі вихідного індивідуального завдання, наведеного у ЛР №1.

Стосовно програмного задавача – див. лекц. матеріал МПС-2.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

8 ЛАБОРАТОРНА РОБОТА № 8

Тема: WEB-диспетчеризація

Мета: Вивчення технологій WEB-диспетчеризації в задачах побудови АСК та моніторингу.

Завдання:

1) Виконати розробку WEB-інтерфейсу власної АСК.

2) Виконати контрольне завдання до ЛР №8.

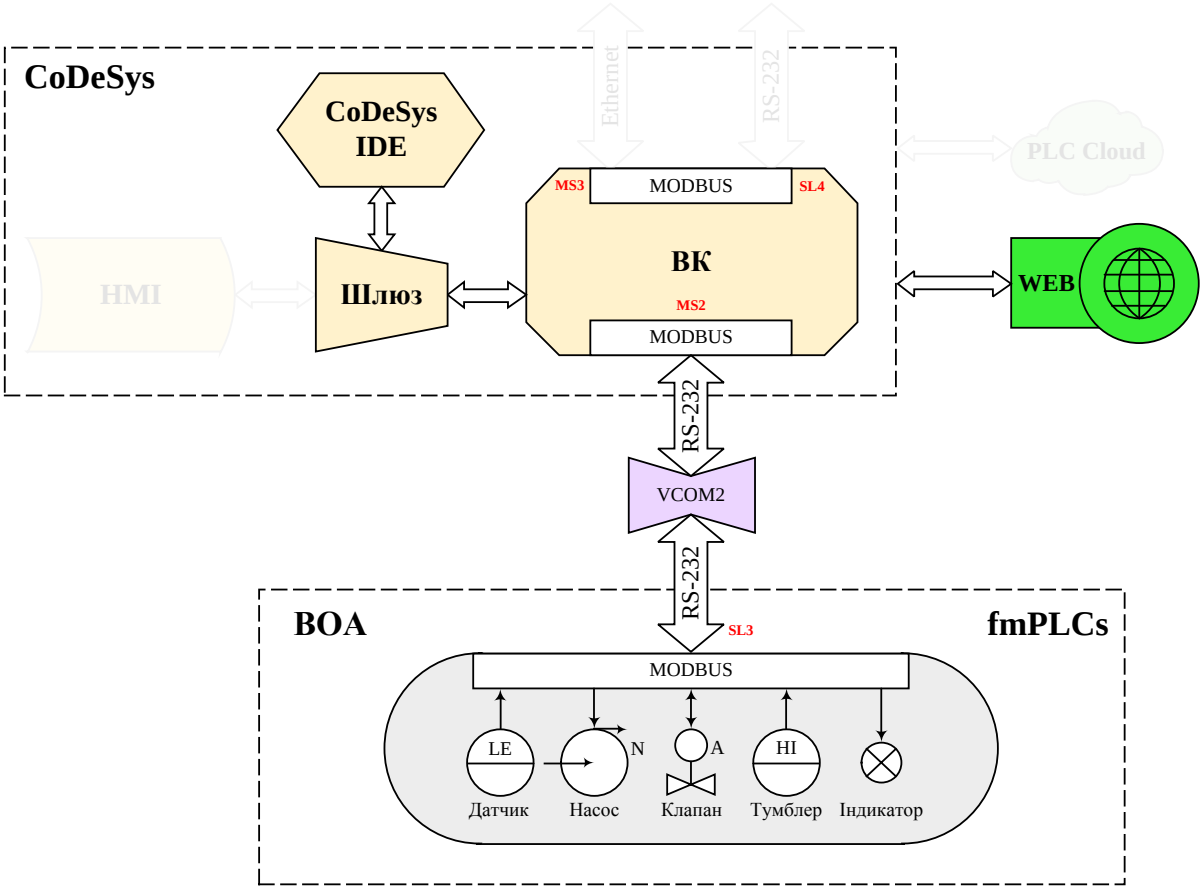
8.1 Постановка задачі розробки WEB-панелі

1) Побудувати HMI-об'єкт засобами CoDeSys.

2) Додати в проект CoDeSys об'єкт WEB-візуалізації.

3) Перенести HMI-об'єкт до WEB-візуалізації.

4) Перевірити працездатність роботи WEB-інтерфейсу АСК.



8.2 Теоретичні відомості

Методологія розробки WEB-інтерфейсів базується на знанням, уміннях і досвіду, що отримані студентом при вивченні дисципліни МПС.

8.3 Приклад виконання роботи

Для відображення візуальних об'єктів в браузері потрібно взяти за основу візуалізацію і менеджер візуалізації з ЛР №5 ПрІС-1 скопіювати їх та додати до поточного проекту в «**Application**». Додати до проекту WEB-візуалізацію за допомогою менеджера візуалізації. Для цього потрібно кліком ПКМ на «**Менеджер визуализации**» додати об'єкт «**Web-визуализация**» в менеджер візуалізації та задати йому ім'я.

Елемент «**Web-визуализация**», встановлений під «**Менеджером визуализации»,** містить установки для передачі цільової візуалізації по Internet**.** Їх можна змінити у вікні редактора, який відкривається подвійним натисканням на об'єкті:

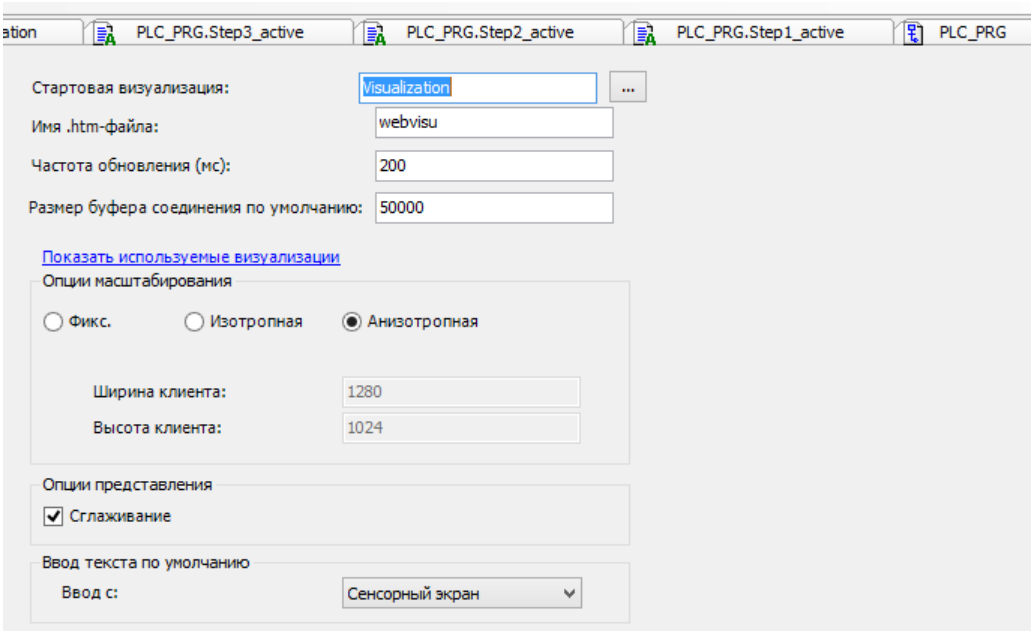


Рис. 1. Налаштування об'єкта WebVisualization

У властивостях об'єкту «**WebVisualization**» потрібно задати початкову візуалізацію для відображення в браузері «**Start Visualization→Visualization**», назва файлу відображення (за замовчуванням **webvisu**), період обновлень (за замовчуванням 200мс) і максимальний розмір буфера (за замовчуванням 50000 байт). В нижній частині «**Best Fit**» налаштовуються параметри відображення екрана візуалізації.

WEB-візуалізація реалізується у вигляді Java-додатку, який при запуску запитує інформацію для відображення з WEB-сервера. Після цього тільки зміни, що відображаються передаються в циклічному режимі. При загрузці проекта візуалізації в піддиректорію /visu цільової системи передаються всі необхідні файли для WEB-візуалізації. До них відносяться Java-додаток, базова HTML-сторінка (*.htm-файл) візуалізація і всі використовувані зображення.

Перед викликом WEB-візуалізації потрібно встановити на ПК Java (наприклад із сайта http://www.java.com/ru/).

Для виклику WEB-візуалізації через Internet потрібно в браузері ввести наступний адрес:

```
http://<IP-адрес web-сервера>:<порт web-сервера>/<webvisu>.htm
```

В даному прикладі він має вигляд:

```
http://localhost:8080/webvisu.htm
```

webvisu.htm - це htm-файл, заданий в «**Менеджере визуализации»**, за допомогою якого у вікні браузера буде показано стартова візуалізація, яка також задана в «**Менеджере визуализации».** Після цього з візуалізацією можна працювати в самому браузері, як показано на рисунку.

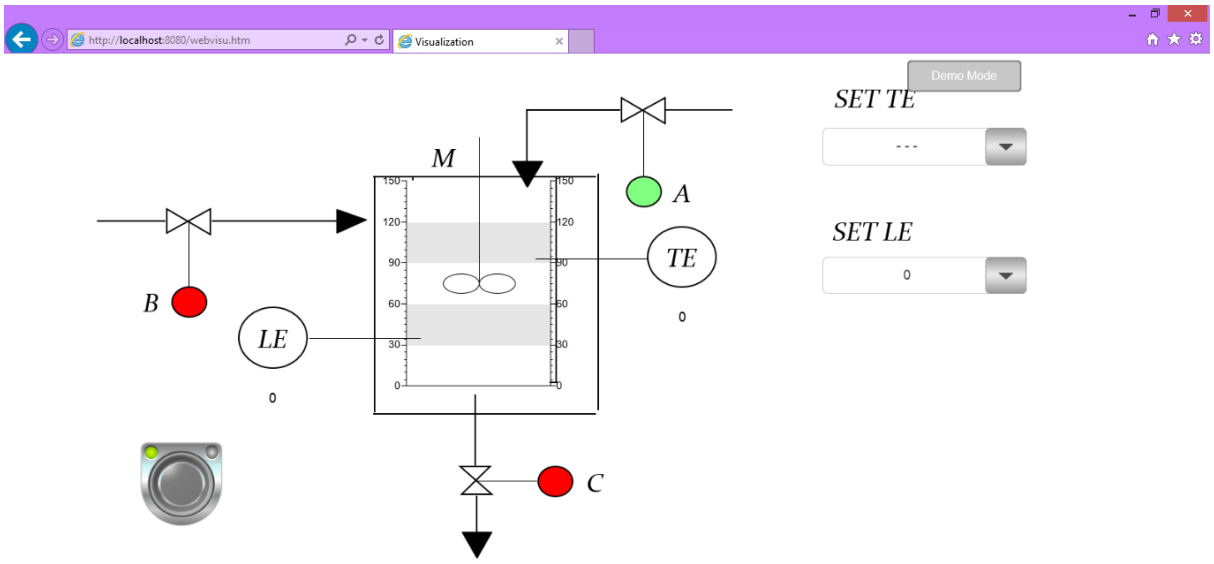


Рис. 2. Робота с Web-візуалізацією в браузері

Для додавання WEB-візуалізації у CoDeSys версії 2 потрібно впевнитись у підтримці WEB-сервера на ПЛК. Перед викликом WEB-візуалізації потрібно встановити на ПК Java (наприклад з сайта http://www.java.com/ru/).

Створюємо новий проект і в якості контролера візьмемо dk55, оскільки він підтримує WEB-сервер («**Target Settings**» в категорії «**Configuration**» вибрати «**DK55_Full_FP_CM**»), можна також обрати будь-який інший контролер що підтримує WEB-сервер. В налаштуваннях таргета звертаємо увагу на галочку «**Web-Visualization**» на вкладці «**Visualization**» – вона повинна бути обов'язково.

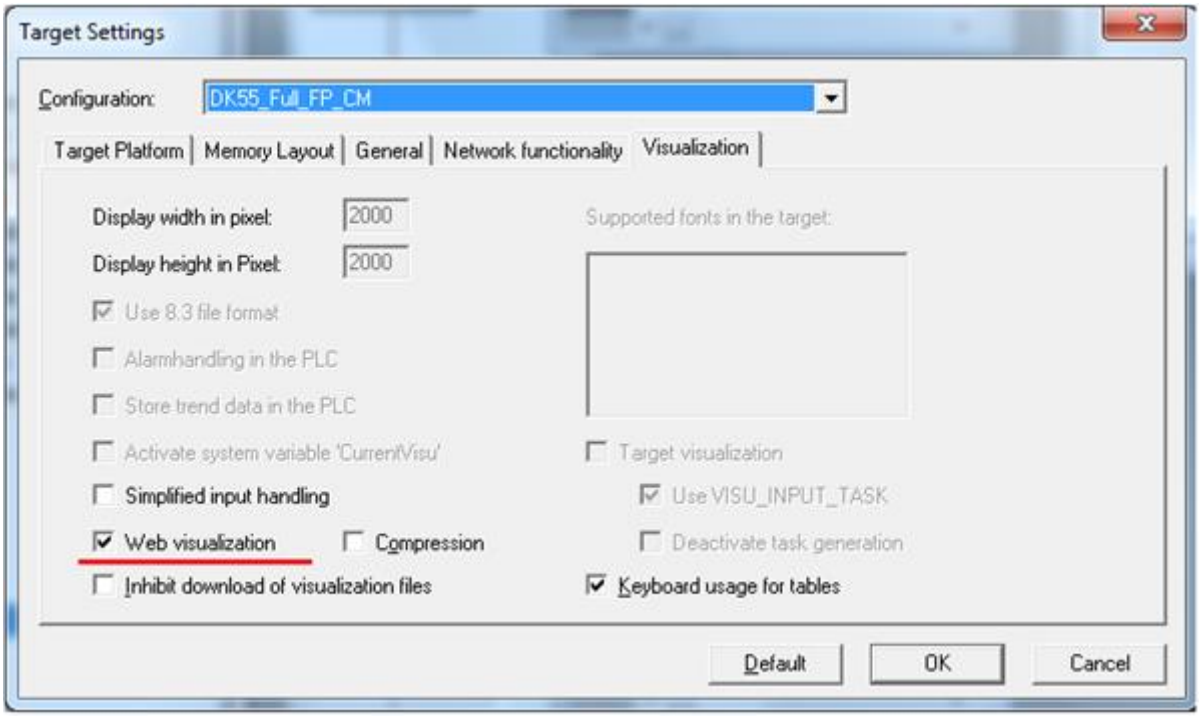


Рис. 3. Додавання WebVisualization в проект CoDeSys v.2

Цією галочкою ми дозволяємо загружать веб-візуалізацію в контроллер при компіляції.

У вкладці «**Visualizations**» створюємо нову візуалізацію **PLC_VISU** (обов'язково з такою назвою!) і переносимо візуальні об'єкти з проекта що був створений в ЛР №5 ПрІС-1 та прикріплюємо їх значення до змінних програми.

Якщо під час відображення в браузері будуть скачки або завмирання якихось значень то треба зменшити швидкість розрахунків у програмі тим самим зменшимо завмирання.

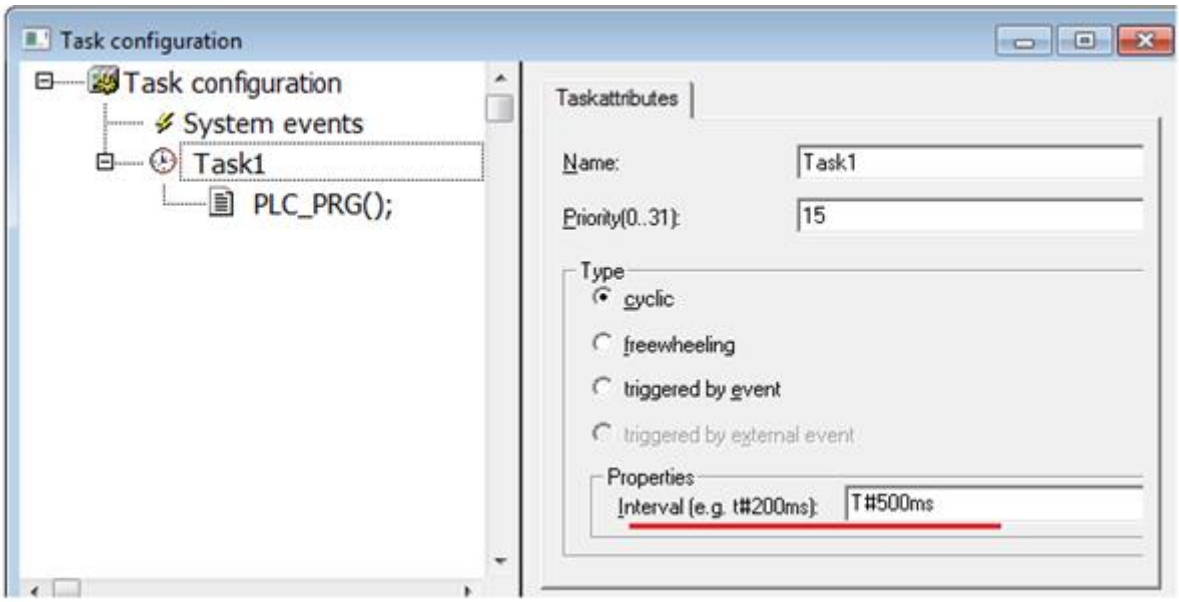


Рис. 4. Налаштування параметрів програми CoDeSys v.2

Це дозволить розраховувати значення змінних раз в пів секунди.

При підключення до контролера відбудеться компіляція проекта і загрузка файлів візуалізації в контролер. Після запуску проекта на виконання можемо переходити безпосередньо в браузер і управляти об'єктом із нього.

Запускаємо браузер і в адресному рядку вводимо

```
http://<IP-адрес web-сервера>
```

Якщо в текстовому полі введені кириличні символи, то в браузері вони не будуть відображатись. Для того щоб можна було їх відображати потрібно додати в файл **webvisu.htm** параметр

`param name="DEFAULTENCODING" value="TRUE".`

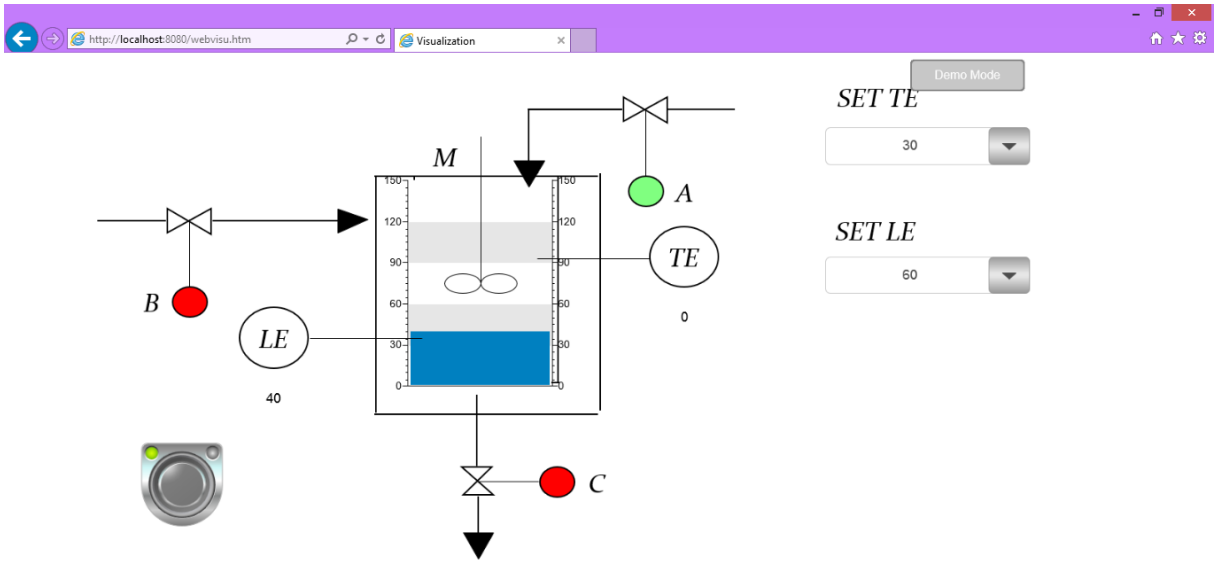
Краще це зробити в оригінальному файлі

C:\Program Files\3S Software\CoDeSys V2.3\Visu\webvisu.htm.

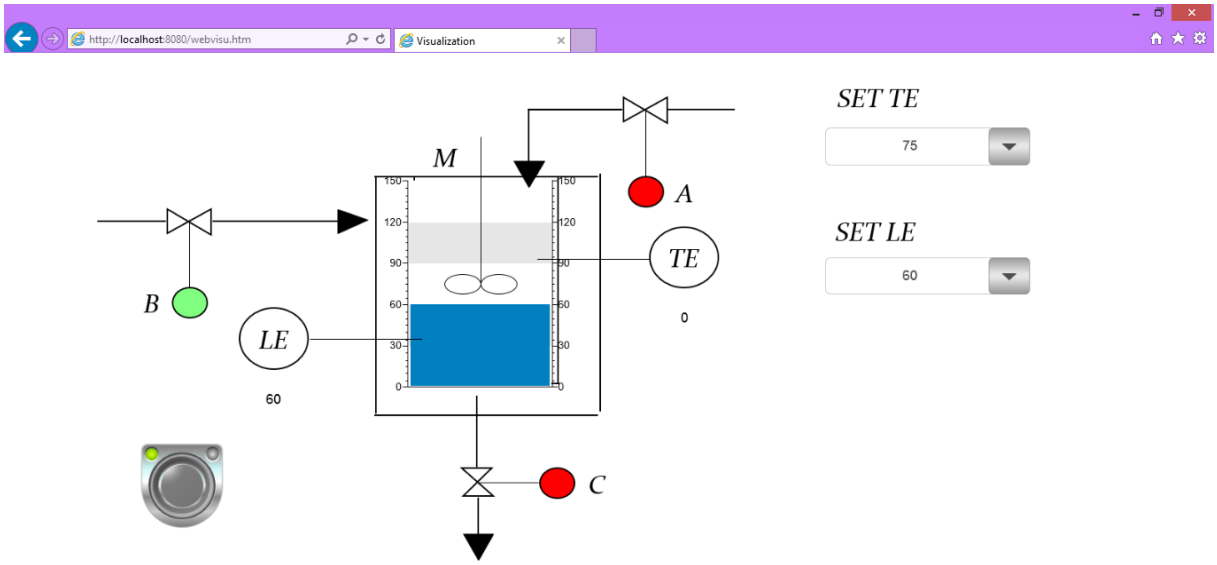
```
…
</HEAD>
<BODY onload=”CookieHandling()”
<APPLET CODEBASE=. CODE=WebVisu.class name=”WebVisu”with
<param name=”archive” value=”webvisu.jar,minm1.jar”>
<param name=”DEFAULTENCODING” value=”TRUE”>
</APPLET>
</BODY>
</HTML>
```

8.4 Приклад роботи WEB-візуалізації у браузері

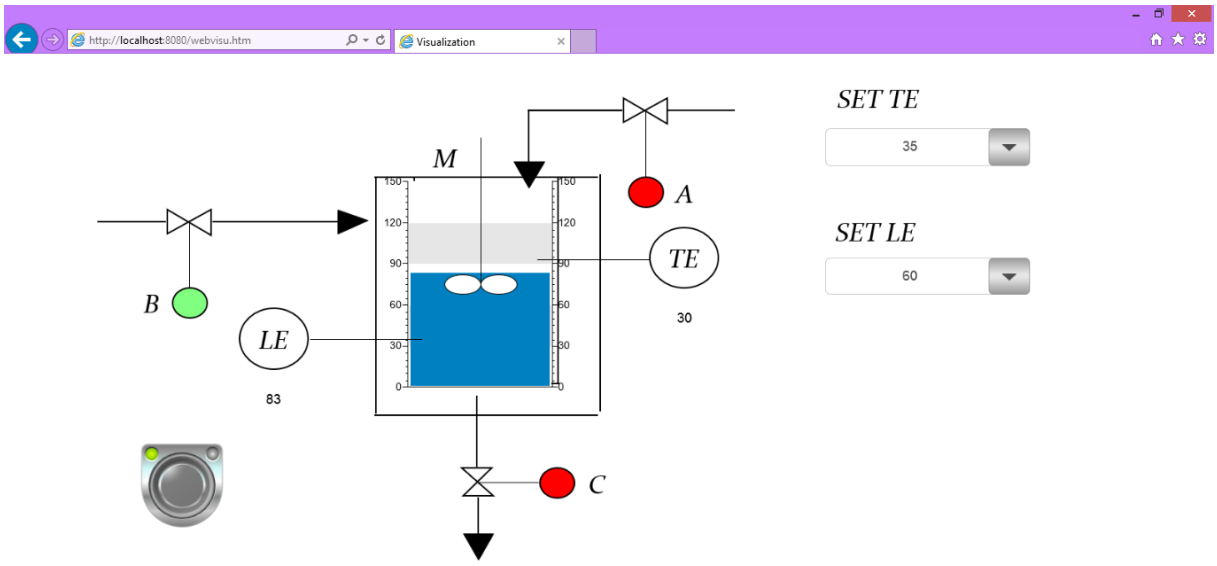
Включаємо подачу холодної води (клапан **A**):



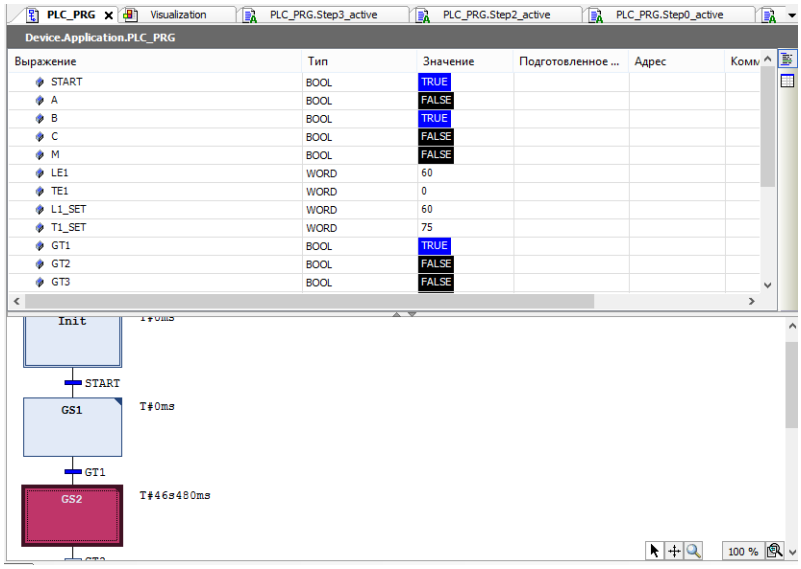
Рівень води у ємності збільшується і коли досягає встановленого значення подача холодної води відключається і включається подача гарячої води (клапан **B**):



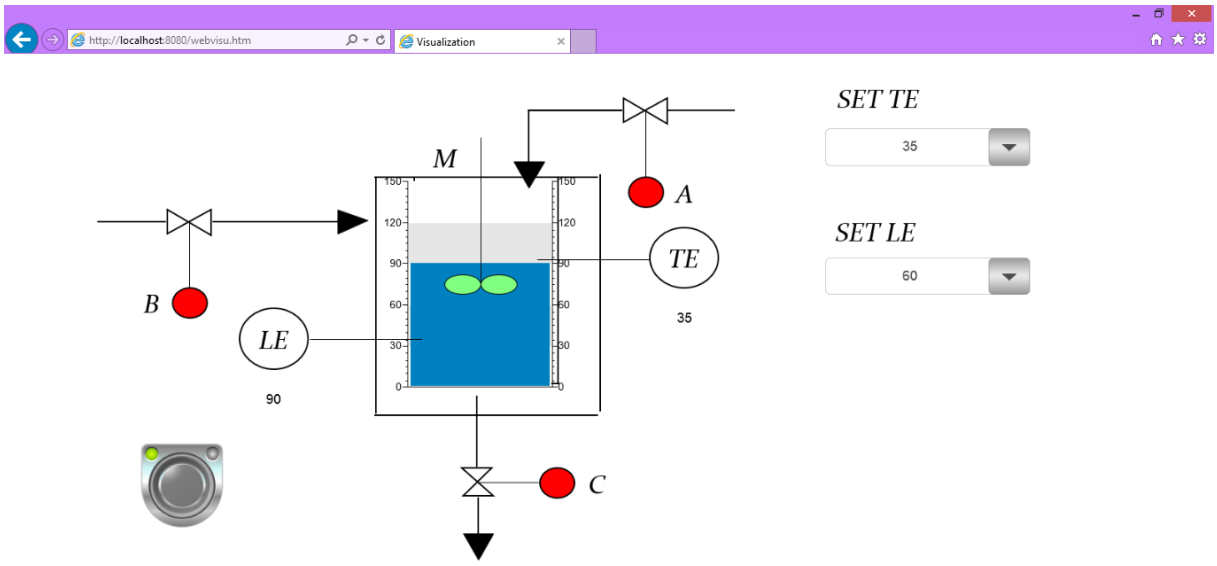
Гаряча вода теплоносія поступає, рівень води у ємності збільшується і температура компоненту зростає:



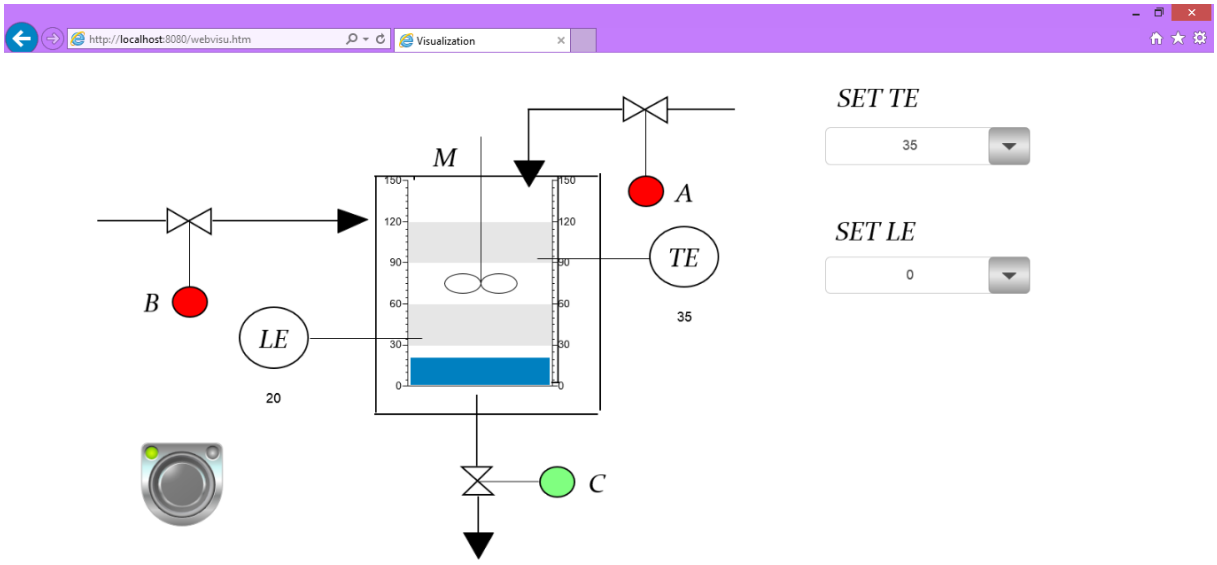
Оновлення значень змінних на стороні CoDeSys:



Коли температура досягає заданого рівня відключається подача гарячої води і включається мішалка **M** на декілька секунд:



Після виключення мішалки відкривається стічний клапан **C** і вода починає витікати з ємності:



8.5 Контрольні запитання

1) Як в пакеті CoDeSys створити візуалізацію?

2) Як в проект додати веб-візуалізацію?

3) Який принцип відображення веб-візуалізації?

4) За допомогою якого аплету відбувається постійне оновлення даних?

5) З чого складається адреса для відображення в браузері веб-візуалізації?

8.6 Контрольне завдання

Розробити WEB-HMI на базі вихідного індивідуального завдання, наведеного у ЛР №1.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

9 ЛАБОРАТОРНА РОБОТА № 9

Тема: Розробка АРМО в Trace Mode та його зв'язок із ВК

Мета: Вивчення базових можливостей проектування АРМО засобами інструментального середовища Trace Mode на практичному прикладі АСК.

Завдання:

1) Ознайомитись із середовищем Trace Mode для розробки АРМО.

2) Засвоїти методику та послідовність розробки АРМО в середовищі Trace Mode шляхом створення: бази каналів; джерел сигналів із MODBUS-мережі; прив'язки каналів до джерел; Екранних форм візуалізації АСК та Тренду роботи АСК.

3) Виконати контрольне завдання до ЛР №9.

9.1 Постановка задачі розробки АРМО

1) Використати за базу робочий вихідний проект із ЛР №2-5;

2) Перевірити наявність на ПК двох пар віртуальних портів RS-232, за умови їх відсутності – встановити та знайти їх номери, відповідно;

3) Додати до проекту CoDeSys **SL**-канал **MODBUS** та COM-порт;

4) Налаштувати параметри передачі-прийому у відповідністю із таблицею змінних проекту. Сформувати таблицю регістрів **MODBUS**;

5) Створити проект АРМО засобами Trace Mode;

6) Додати до проекту Trace Mode **MS**-канал **MODBUS** та COM-порт;

7) Налаштувати параметри передачі-прийому у відповідністю із таблицею змінних проекту. Сформувати таблицю каналів Trace Mode;

8) Додати до Екрану АРМО компонент «**Тренд**» та налаштувати канали для відображення змінних на Тренді;

9) Завантажити модифікований проект CoDeSys до ВК;

10) Виконати запуск емуляції роботи АРМО АСК, змінюючи значення давачів у ВОА.

Лабораторний стенд на базі ПК складається із наступних елементів:

1) Середовище програмування CoDeSys;

2) ВК – **CoDeSys Control Win**;

3) Інструментальне середовище Trace Mode для розробки АРМО.

4) Віртуальний RS-232 – «**ELTIMA Virtual COM**» – VCOM;

5) **ВОА – автоматичний симулятор** ОК + ТЗА – **fmPLCs + скрипт**;

Структура стенду наведена на рис. 6.1.

Якщо на ПК встановлено віртуальний COM-порт, необхідно обрати «**Мій комп'ютер → Властивості → Диспетчер пристроїв**» та знайти номери віртуальних COM-портів інтерфейсу ELTIMA парами. На рис. 6.1 наведений приклад коли номери віртуальних COM-портів є COM1+COM2 та COM3+COM4. Вони віртуально та попарно **з'єднані** (аналог фізичного з'єднання кабелем).

Функціонування стенду №6 наступне. Прикладна програма CoDeSys, завантажена у ВК, через інтерфейс віртуального COM1 (**MS2**) передає команди протоколом **MODBUS**. Віртуальним каналом зв'язку команди надходять до віртуального COM2 (**SL2**), який у свою чергу передає команди до ВОА, що надходять до виконавчих пристроїв АСК. В свою чергу ВОА повертає до ВК необхідні дані з давачів про поточний стан АСК. Аналогічним чином АРМО (**MS1**) формує команди до ВК (**SL1**) та отримує дані у відповідь про стан роботи АСК.

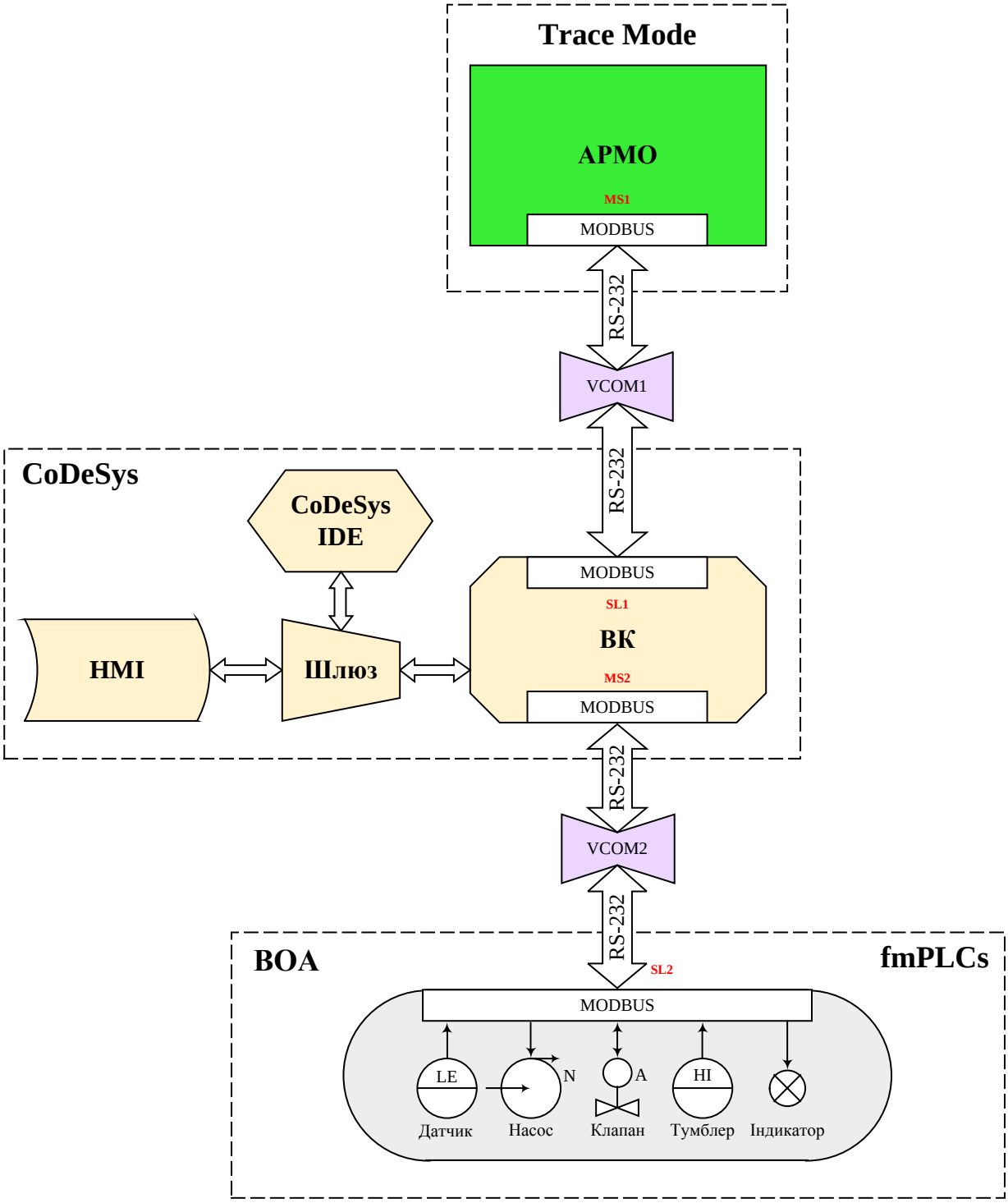


Рис. 6.1. Структура віртуального лабораторного стенду №6

9.2 Базові теоретичні відомості про Trace Mode

Trace Mode (вимовляється «Трейс Моуд») – інструментальний програмний комплекс класу SCADA HMI, розроблений компанією **AdAstrA Research Group** (Москва) в 1992 році. Призначений для розробки програмного забезпечення АСУТП, систем телемеханіки, автоматизації будівель, систем обліку електроенергії, води, газу, тепла, а також для забезпечення їх функціонування в реальному часі. Починаючи з версії 4.20 (1995 рік) Trace Mode має функції для програмування ПЛК.

Trace Mode складається з інструментальної системи і з набору виконавчих модулів (**runtime**'s). За допомогою виконавчих модулів Trace Mode проект АСК запускається на виконання в реальному часі на автоматизованому робочому місці диспетчера або оператора (АРМО).

Особливістю Trace Mode є «**технологія єдиної лінії програмування**» тобто можливість розробки всіх модулів АСК за допомогою одного інструментарію. Технологія єдиної лінії програмування дозволяє в рамках одного проекту створювати засоби HMI, системи обліку ресурсів, програмувати ПЛК і розробляти WEB-інтерфейс. Для цього в інструментальну систему вбудовані спеціалізовані редактори Trace Mode:

- Редактори графічних мнемосхем та Екранних панелей;
- Редактор програм на мовах стандарту МЕК 6-1131/3;
- Редактор шаблонів й зв'язків із СУБД;
- Редактор паспортів обладнання (EAM);
- Редактор персоналу (HRM) та матеріальних ресурсів (MES).

9.3 Приклад виконання роботи

Послідовність налаштування зв'язку між ВК та ВОА докладно наведена у ЛР №4.

У якості базового проекту до цієї ЛР взяти проект із ЛР №5.

ЗАУВАЖЕННЯ. Старт АСК у ЛР №2-5 виконується віртуальною кнопкою «**СТАРТ**» (рівень ВОА). У наведеному прикладі на рівні АРМО відображатиметься **лише стан цієї кнопки**. Задля можливості керувати АСК в режимі «СТАРТ-СТОП» з рівня АРМО потрібно самостійно до АРМО додати окрему кнопку (наприклад, «**ЗАПУСК**»), канал для неї та відповідний мережевий зв'язок із нижнім рівнем ВК.

Перед початком роботи необхідно вдосконалитись у наявності, окрім створених у ЛР №2-5 **віртуальних** COM-портів (1 та 2), додаткової пари COM-портів (3, 4) на ПК (у випадку відсутності – створити).

Для організації інформаційного зв'язку між ВК та АРМО SCADA-системи Trace Mode за допомогою протоколу MODBUS, треба в CoDeSys додатково додати Slave COM-пристрій, оскільки ВК буде виконувати роль веденого пристрою (**SL**), а АРМО – ведучого (**MS**). Для цього потрібно використати проект CoDeSys з ЛР №5 (рис. 6.2).

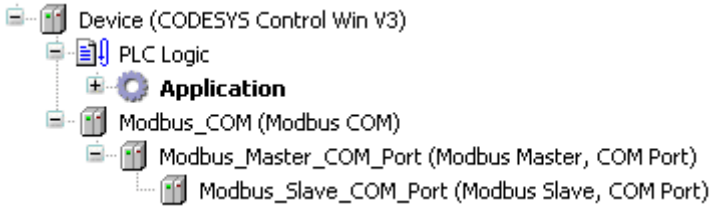


Рис. 6.2. Дерево пристроїв у Навігаторі проекту CoDeSys

Спочатку потрібно виконати «**Devices → Add Device**» і додати другий об'єкт **Modbus_COM (**мережу для зв'язку ВК і АРМО**)**, а потім, натискаючи ПКМ на ньому, додати новий SL-пристрій (див. рис. 6.3).

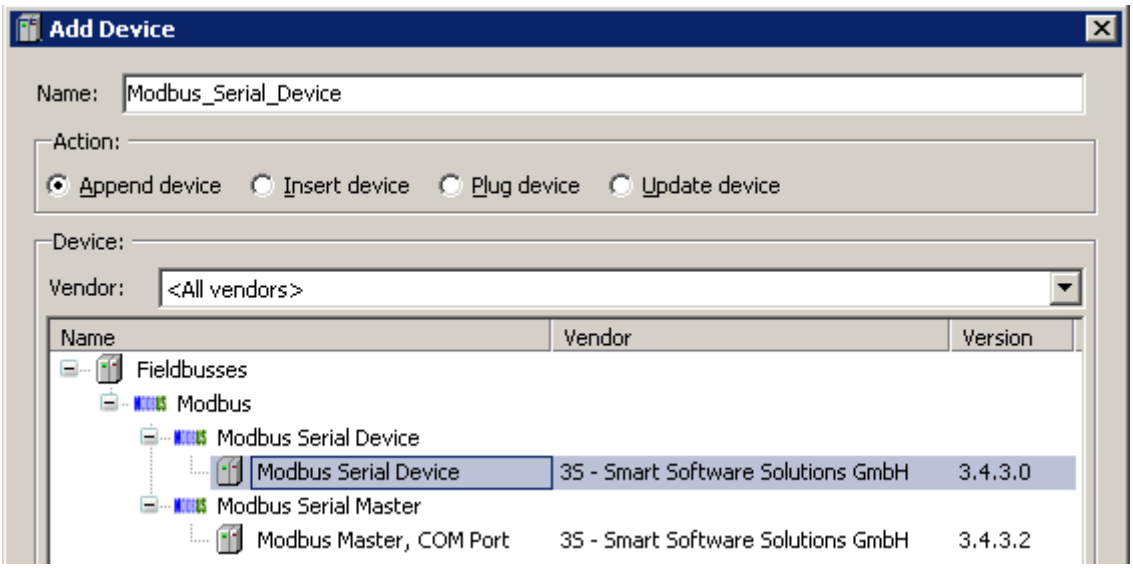


Рис. 6.3. Діалог додавання SL-пристрою

Після додання SL-пристрою, потрібно виконати для нього конфігурацію COM-порту, пам'ятаючи, що емульовані COM-порти (ELTIMA Virtual) на ПК **віртуально поєднані парами**, тобто COM1 зв'язаний лише з COM2, COM3 з COM4 тощо. Отже, треба задати параметри COM-порту SL-пристрою, як показано на рисунку 6.4.

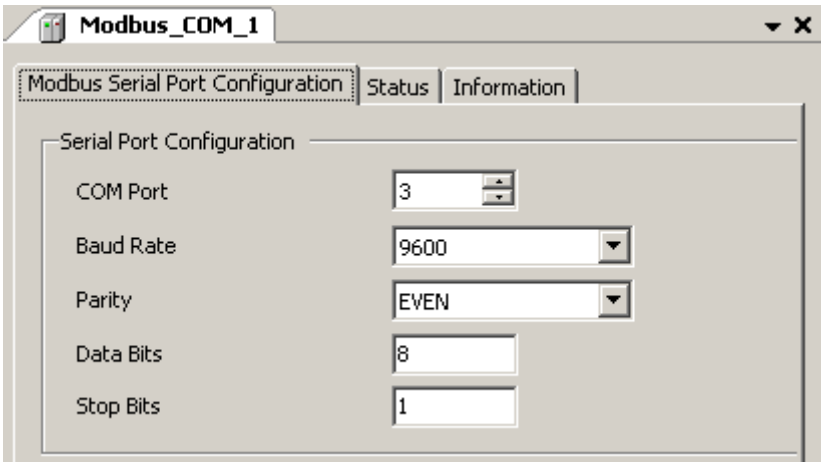


Рис. 6.4. Конфігурація COM-пристрою

Після налаштування COM-порту SL-пристрою треба задати налаштування самого MODBUS-пристрою: номер SL-пристрою на шині (ввести адресу 1) та кількість MODBUS-каналів введення/виведення (рис. 6.5). Число

Holding Registers визначає довжину приймального буферу даних від АРМО, а **Input Registers** – передаючого буферу до АРМО.

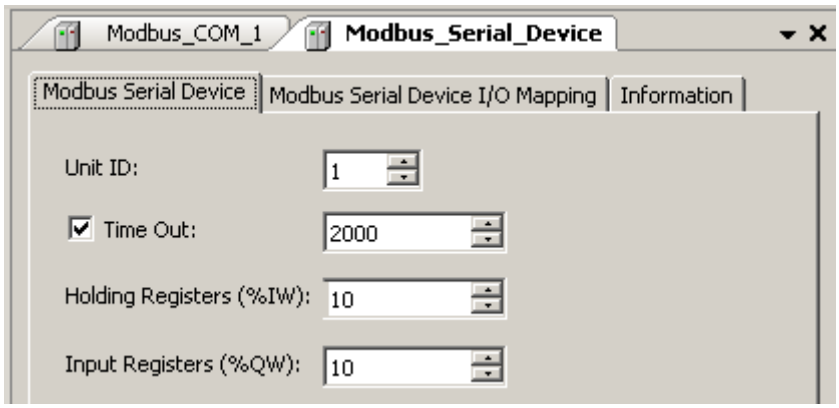


Рис. 6.5. Налаштування MODBUS SL-пристрою

Після цього, як і у ЛР №4, потрібно задати маппінг (відповідність) каналів до змінних програми ВК. Для того, щоб встановити відповідність для **Holding Registers** (тип даних WORD), треба використовувати наявні в таблиці маппінгу рядки, не розкриваючи їх (рис. 6.6).

Масиви Inputs[] та Outputs[] є окремим сегментом (буфером) розміщення даних, що призначений для обміну по шині MODBUS. До масиву Inputs[] прив'язуються програмні змінні ВК, у які ВК буде приймати дані від АРМО, а до масиву Outputs[] прив'язуються змінні, дані з яких будуть передаватись від ВК до АРМО.

Для того ж, щоб задати передачу даних виду **Coils** (тип даних BOOL, окремі біти) протоколом MODBUS, треба натиснути символ «+», щоб розкрити рядки таблиці, та прив'язати необхідні булеві змінні до відповідних бітів (рис. 6.6).

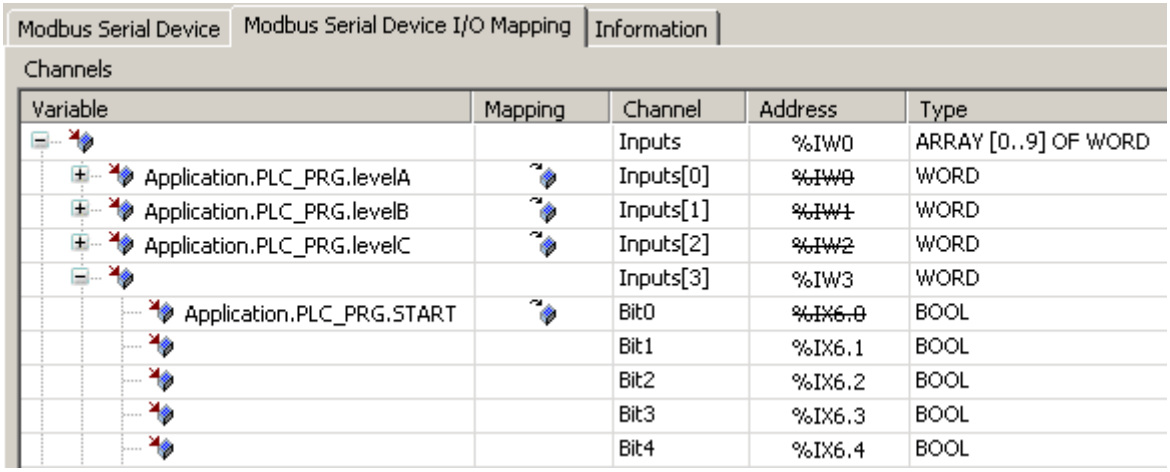


Рис. 6.6. Налаштування каналів введення даних від АРМО до ВК

Для прийому-передачі даних типу REAL (FLOAT) необхідно в цей буфер задавати подвійне слово, що складатиметься із молодшого та старшого слів, розташованих один за одним.

ЗАУВАЖЕННЯ. За наявності проблеми скидання значень мережевих параметрів в нуль, що передаються АРМО до нижнього рівня ВК, у налаштуваннях «**Modbus Serial Device**» необхідно вимкнути параметр «**Time Out**» (рис. 6.5) шляхом зняття галочки.

Аналогічним чином виконується маппінг змінних, що відправляє ВК до АРМО (рис. 6.7).

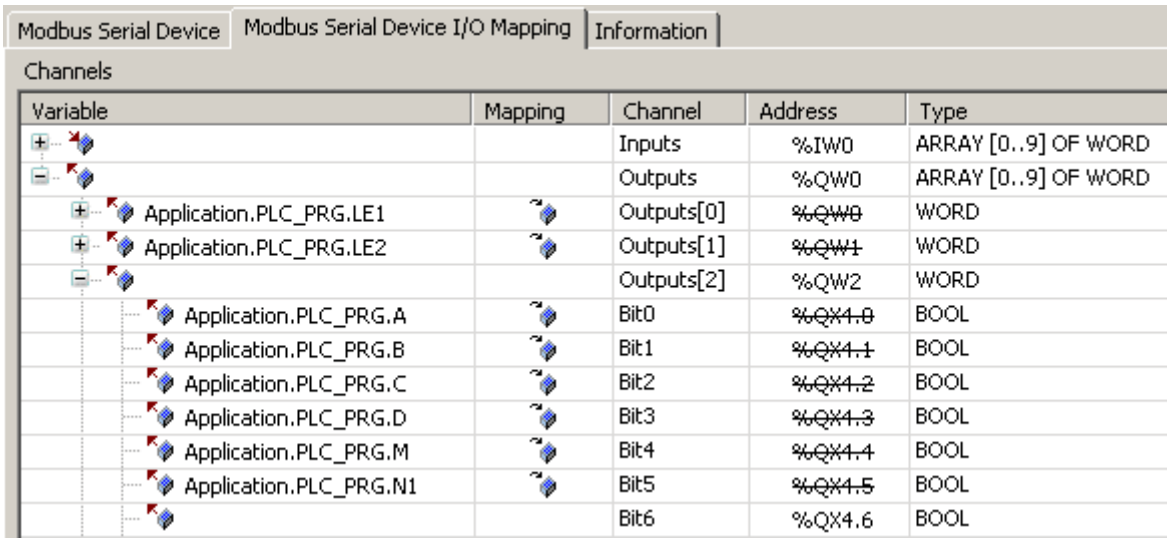


Рис. 6.7. Налаштування каналів виведення даних від ВК до АРМО

На цьому конфігурування ВК в середовищі CoDeSys можна вважати закінченим.

9.4 Приклад розробки АРМО в Trace Mode

Після запуску Trace Mode, треба створити новий проект та нову групу MODBUS в списку «**Источники/Приёмники**» (рис. 6.8).

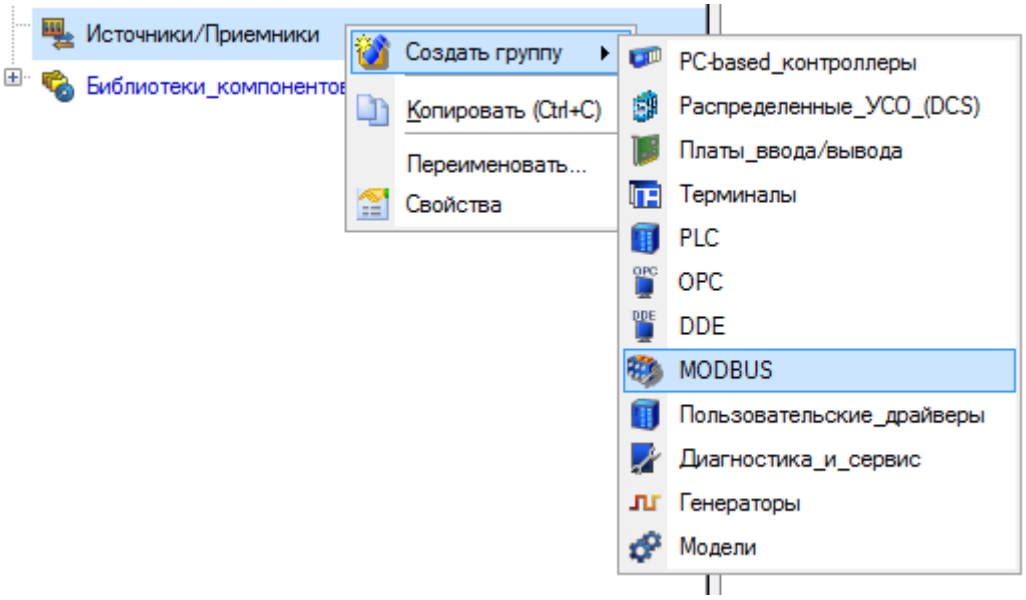


Рис. 6.8. Створення групи MODBUS

Далі потрібно додати до створеної групи MODBUS_1 необхідні канали MODBUS за допомогою контекстного меню, що відкривається шляхом натискання ПКМ на групі (рис. 6.9).

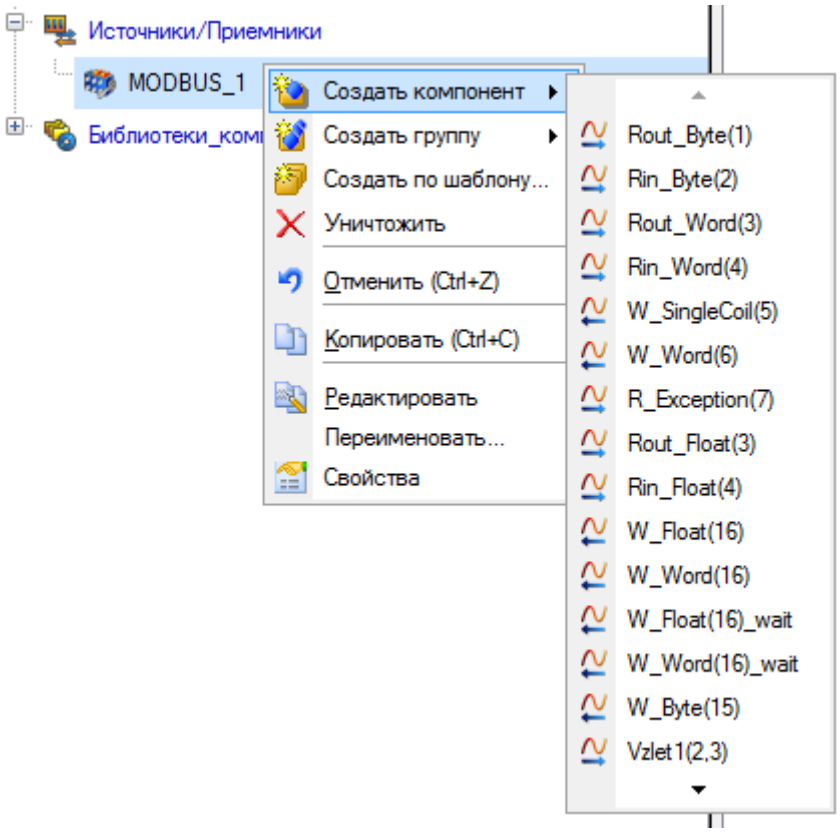


Рис. 6.9. Додавання каналів до групи MODBUS

В термінології Trace Mode треба використати два типи каналів: **Rin_Word(4)** та **W_Word(6)** – зчитування та запис даних відповідно.

ЗАУВАЖЕННЯ. Ці обидва канали MODBUS використовуються як для зчитування/запису слів, так і для зчитування/запису окремих бітів. У дужках в назві каналів вказаний код функції (4 та 6) протоколу MODBUS.

Канали **levelA**, **levelB**, **levelC** використовуються для передачі уставок з АРМО до ВК, отже їх тип **W_Word(6)**. Тож треба створити спочатку канал (рис. 6.10) для змінної **levelA** і проконфігурувати його.

В налаштуваннях задати ім'я каналу, кодування не змінювати. Далі в полі «**Номер порта**» для 4-го COM-порту вказати значення **0x3**. У полі «**Адрес**» вказати номеру SL-пристрою (див. налаштунки проекту в CoDeSys) – **0x1**. У полі «**Направление**» обирається **Output** оскільки каналом передаються дані від ВК до АРМО. Формат залишити дискретним (для типу даних WORD).

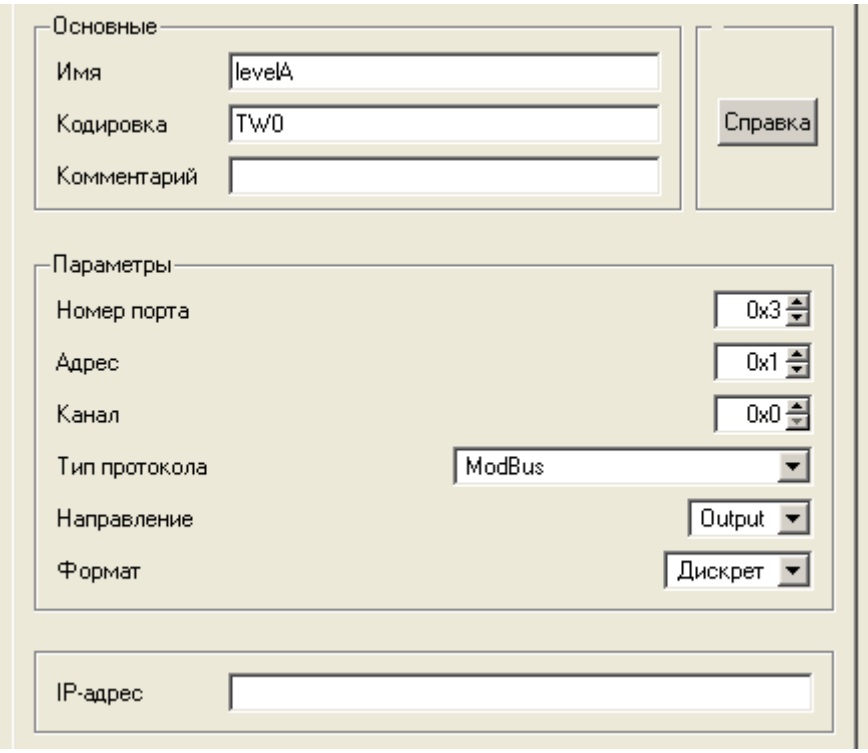


Рис. 6.10. Налаштування каналу levelA

В полі «**Канал**» задається зміщення шини MODBUS у відповідності із маппінгом в CoDeSys (див. рис. 6.10). Змінна levelA прив'язана до Inputs[0], тож зміщення **0x0**. Канали levelB, levelC прив'язані до Inputs[1] та Inputs[2] відповідно й конфігуруються аналогічним чином.

Далі треба додати решту каналів (рис. 6.11), які потрібні для функціонування АСК, спираючись на маппінг в CoDeSys (див. ЛР №2).

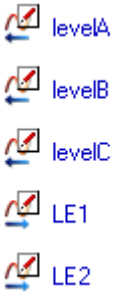


Рис. 6.11. Деякі MODBUS-канали джерел даних в Trace Mode

Канали LE1 та LE2 (давачі рівнів) налаштовуються аналогічним чином, але напрямок задається **Input** і зміщення згідно маппінгу.

В той самий спосіб налаштовуються канали для передачі **Coils** (виконавчі пристрої), оскільки вони передаються у складі слів (2-х байт). Для каналів **EXECUTIVE** та **START** (індикація початку роботи АСК) напрямки також **Input**. Але якщо додати до АРМО кнопку глобального запуску АСК, тоді для неї напрям вказується як **Output**.

Далі в проекті Trace Mode шляхом натискання ПКМ на компоненті «**Система**» створюється вузол **RTM_1** (рис. 6.12). Автоматично у цьому вузлі створюється порожня група «**Каналы**».

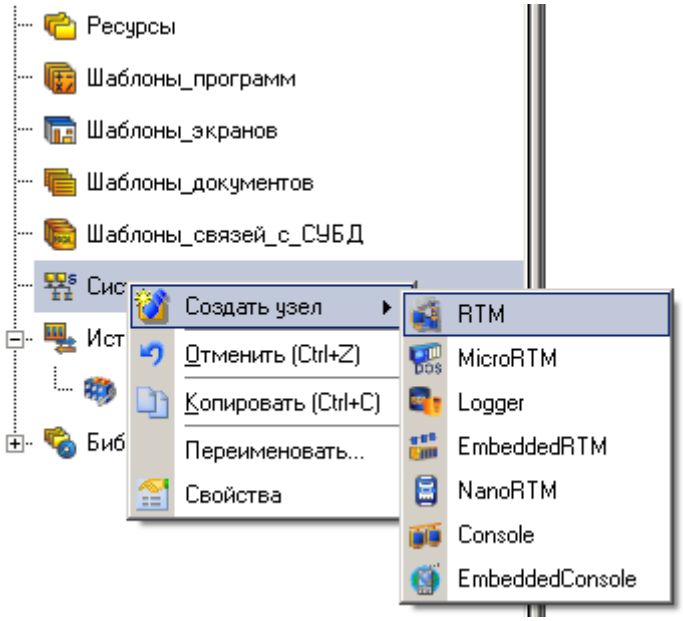


Рис. 6.12. Створення вузла «RTM»

Після завершення конфігурування каналів MODBUS шляхом перетягування (Drag & Drop) треба скопіювати їх до розділу «**Каналы**» під пунктом **RTM_1** в Навігаторі проекту Trace Mode (рис. 6.13).

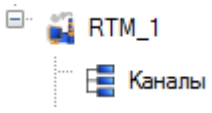


Рис. 6.13. Група «Каналы»

Список каналів в проекті набуде вигляду, як показнано на рис. 6.14.

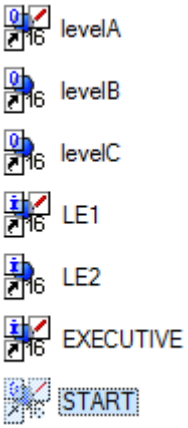


Рис. 6.14. Елементи в групі «Каналы»

Далі під **RTM_1** треба створити групу «COM-порти», а в ній за допомогою контекстного меню створити власне COM-порт (рис. 6.15).

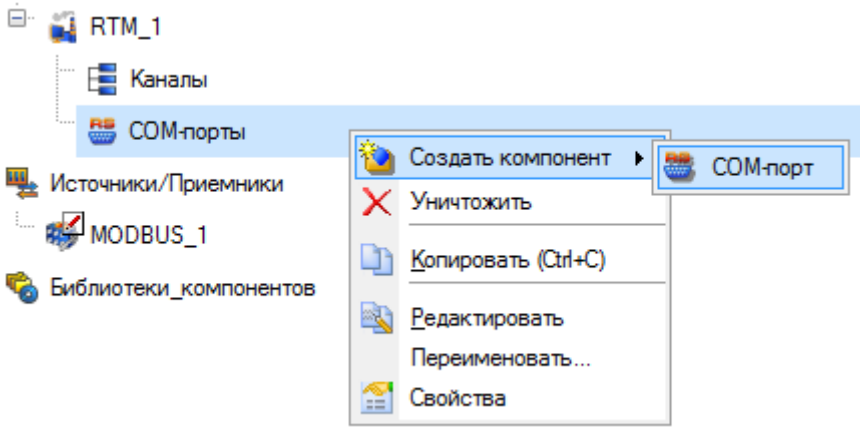


Рис. 6.15. Створення групи COM-портів та самого COM-порту

Подвійний клік ЛКМ на COM-порті відкриє його налаштування, які треба задати у відповідності із рис. 6.16.

ЗАУВАЖЕННЯ. Ці налаштування повинні співпадати з опціями, обраними для COM-порту в CoDeSys, окрім призначення порту: в CoDeSys використовується **Slave**-пристрій (**SL**), а в Trace Mode – **Master** (**MS**).

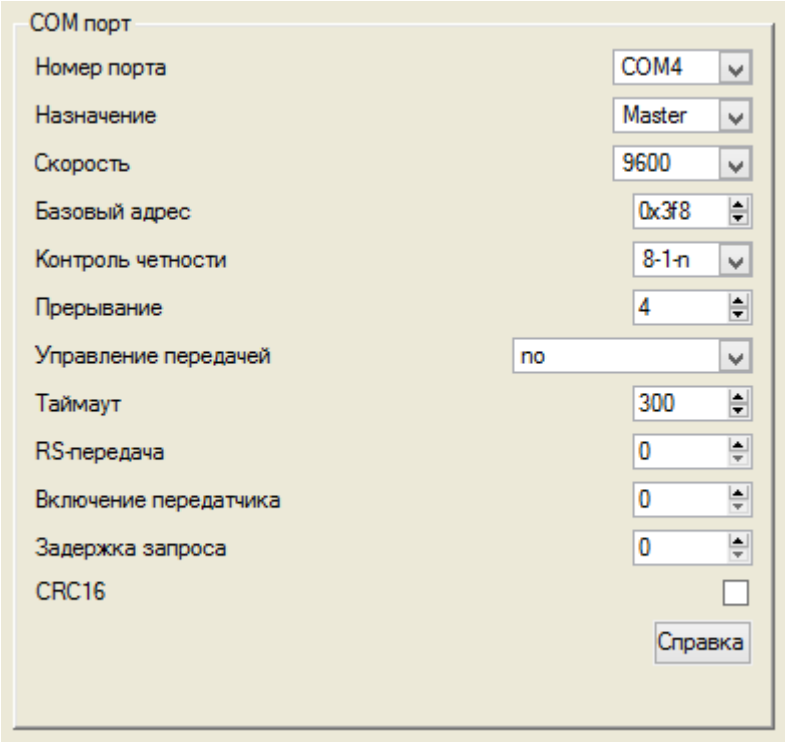


Рис. 6.16. Налаштування COM-порту

Далі необхідно перейти до створення візуалізації АРМО. Для цього треба створити у вузлі **RTM_1** компонент «**Экран**» (рис. 6.17).

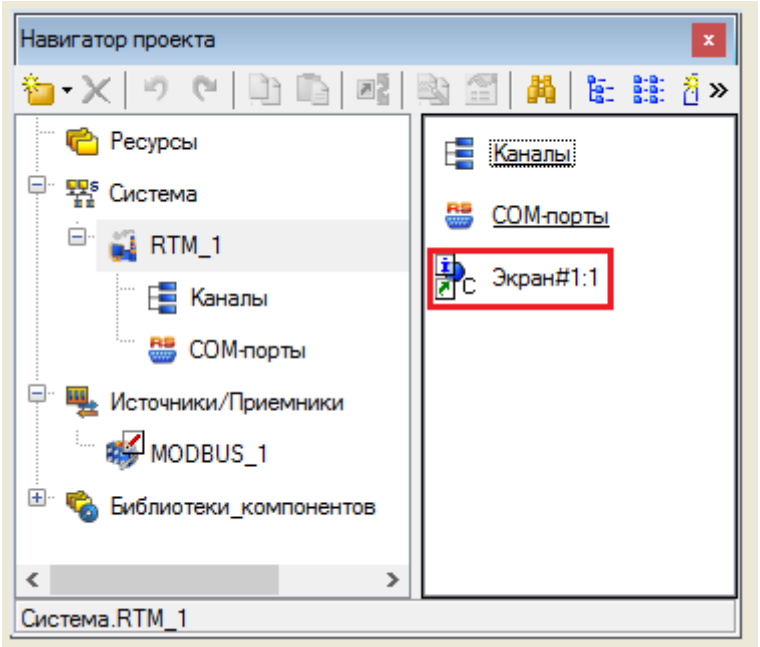


Рис. 6.17. Відкриття Екрану проектування АРМО

Середовище Trace Mode надає широкий спектр інструментів для прототипування АРМО. В розпорядженні користувач має наступний набір компонентів (рис. 6.18), які можна додати до основного простору за допомогою ЛКМ шляхом перетягування (Drag & Drop).

Рис. 6.18. Панель інструментів Trace Mode

Для того, щоб перенести до АРМО задані попередньо канали, потрібно відкрити таблицю каналів («**Меню → Вид → Аргументы экрана**»), та перетягнути (скопіювати та **прив'язати**) канали із групи «**Каналы**» вузла **RTM_1** до цієї таблиці (рис. 6.19).

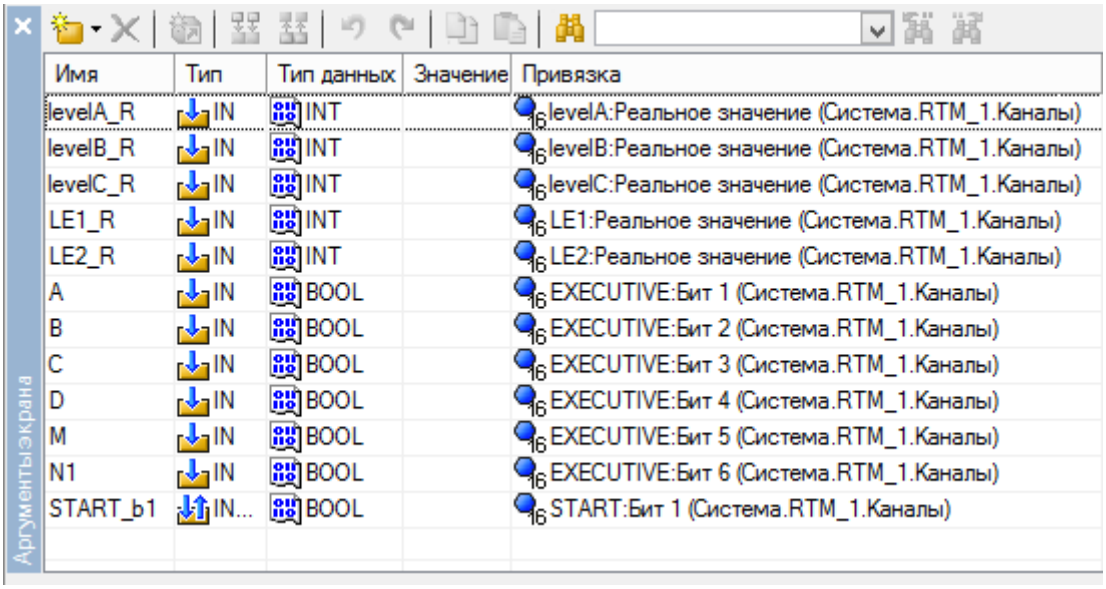


Рис. 6.19. Таблиця аргументів Екрану АРМО

Після цього за допомогою доступних графічних елементів, можна розробити візуалізацію АРМО та проконфігурувати його, як, наприклад, показано на рисунку 6.21.

Канали із таблиці «**Аргументы экрана**» прив'язуються до графічних елементів АРМО також шляхом перетягування (Drag & Drop).

ЗАУВАЖЕННЯ. Для того щоб прив'язати канал із таблиці «**Аргументы экрана**» з графічним елементом «**Экран**» необхідно його виділити в цій таблиці (у стовбці «**Имя**») та, натиснувши ЛКМ й не відпускаючи, перетягнути до необхідного графічного елемента на Екрані АРМО.

Окремо до Екрану АРМО додається так званий «**Тренд**» – графічний елемент для виведення різноманітних графіків. До Тренду підключаються всі необхідні канали проекту також шляхом перетягування (Drag & Drop).

Графічні елементи типу «**Текст**» за замовчуванням налаштовані на відображення статичного тексту. Для налаштування відображення динамічного тексту необхідно змінити це налаштування на відмінне від «**Нет динамизации**».

Для налаштування Кнопки-перемикача «**START**» необхідно використати наступні два поля: «**Вид индикации**» і «**Константа**». Потрібно вибрати в полі «**Вид индикации**» значення «**Arg & Конст**» й встановити в полі «**Константа»** значення «**0x1**». Після цього **Перемикач** почне змінювати значення (TRUE/FALSE) по натисканню.

Для збереження проекту та запуску АРМО потрібно послідовно натиснути виділені кнопки на панелі інструментів (рис. 6.20).

Рис. 6.20. Кнопки запуску проекту

Далі запуститься на виконання монітор реального часу (**МРВ**). В ньому також необхідно натиснути червоного чоловічка.

Після запуску програми на ВК, симулятора ВОА і АРМО в Trace Mode, у Навігаторі проектів CoDeSys можна спостерігати факт успішної комунікації між усіма цими вузлами через MODBUS-мережу (зелені стрілки по колу на рис. 6.22).

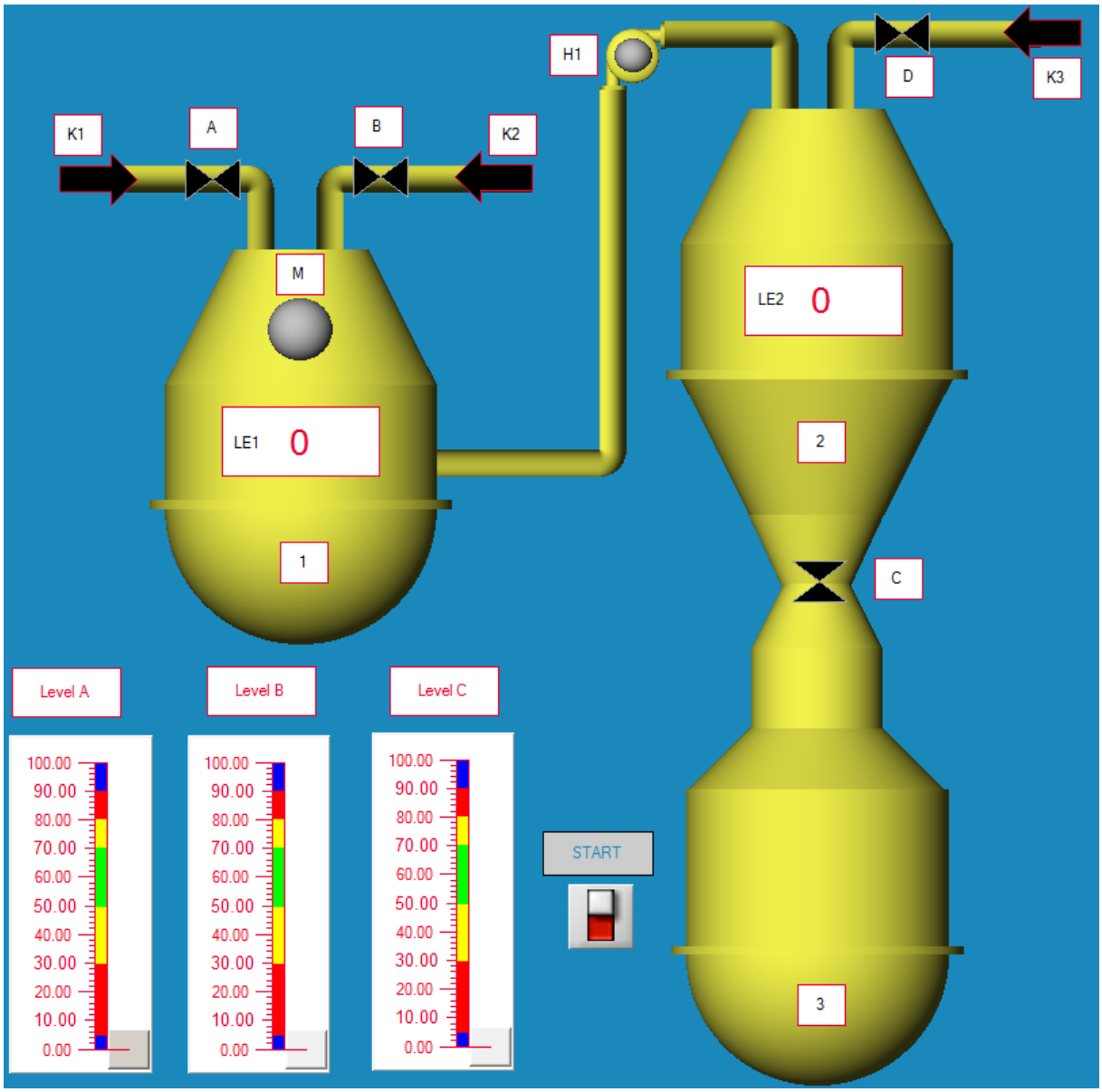


Рис. 6.21. Приклад Екрану АРМО розробленої АСК

Після цього можна спостерігати роботу АСК в інтерактивному режимі: ввімкнення/вимкнення кнопки «**START**», відображення індикаторів, ввімкнення клапанів, побудова графіків на Тренду, тощо.

ЗАУВАЖЕННЯ. У наведеному прикладі АРМО графічний елемент типу тумблер «**START**» на Екрані відображає лише стан (індикація) кнопки «**СТАРТ**», що «фізично» розташована на рівні ВОА.

Проаналізувавши графіки (рис. 6.23), що виводяться на Тренд, можна зробити висновок, що вони співпадають з графіками в **CoDeSys.Trace** середовища CoDeSys (ЛР №3-5).

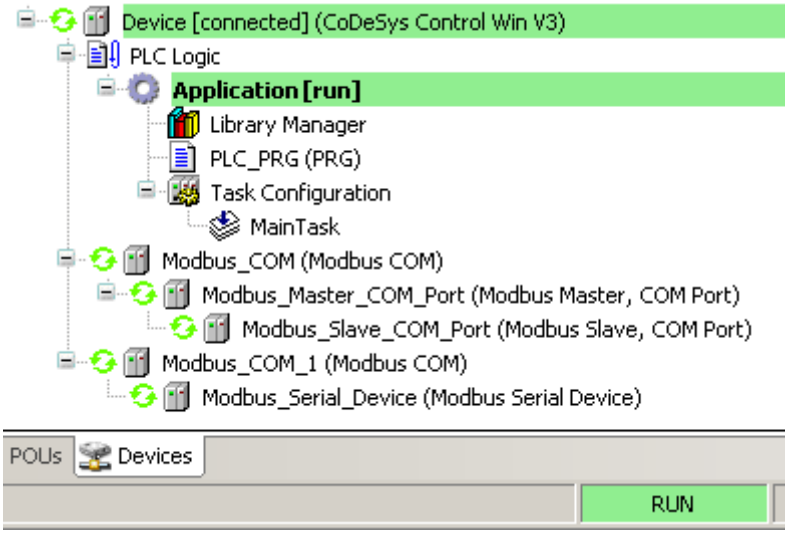


Рис. 6.22. Індикація успішної комунікації ВК, ВОА і АРМО через MODBUS-RS232

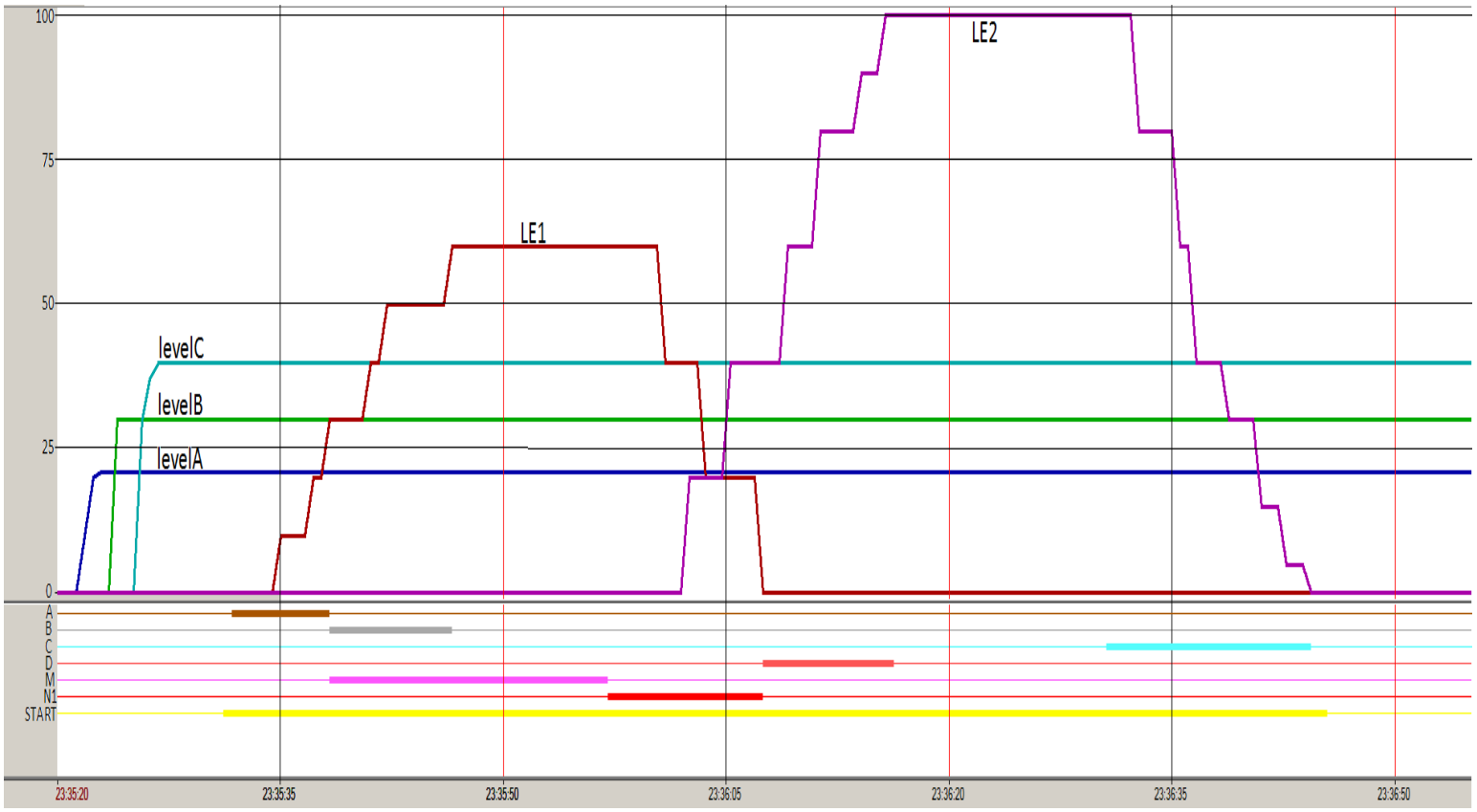


Рис. 6.23. Графіки на АРМО, отримані в ході роботи АСК

9.5 Контрольні запитання

1) Що таке АРМО?

2) Головні особливості Trace Mode.

3) Як створити канал MODBUS в Trace Mode?

4) Прив'язка каналів. Принцип прив'язки.

5) MS та SL. Хто є хто серед CoDeSys та Trace Mode?

6) Як створити Екран та наповнити його каналами.

7) Яким чином канали із джерел попадають до вузла RTM?

8) Як виконати прив'язку аргументів Екрану до графічних елементів?

9.6 Контрольне завдання

1) Використати робочий **Додаток** CoDeSys із ЛР №3-5.

2) Створити додатковий MODBUS-канал типу SL.

3) На АРМО обов'язково створити **окрему КНОПКУ** глобального вмикання АСК та надати їй відповідний функціонал.

4) Виконати емуляцію роботи АСК, запустивши всі вузли (ВОА, ВК, АРМО) на виконання.

Результати ЛР подати у вигляді Екрану АРМО та графіків вхідних й вихідних сигналів в Тренді.

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

10 ЛАБОРАТОРНА РОБОТА № 10

Тема: Взаємодія АРМО із зовнішніми базами даних

Мета: Засвоєння технології інтеграції Trace Mode із зовнішніми реляційними базами даних (БД).

Завдання:

1) Виконати встановлення, налаштування БД у відповідності із таблицями виробничих параметрів й змінних АСК.

2) Створити запити зчитування даних з БД та запису в БД.

3) Виконати контрольне завдання до ЛР №10.

10.1 Постановка задачі взаємодії АРМО із базами даних

1) Встановлення та налаштування зовнішньої СУБД Postgres.

2) Додавання у БД прикладу таблиці змінних АСК.

3) Перевірка підключення СУБД до БД.

4) Налаштування Trace Mode та підключення БД в Trace Mode.

5) Створити SQL-запити до БД та перевірити функціональність.

Струкура лабораторного стенду зображена на рисунку 3.

10.2 Теоретичні відомості про БД

Інформація про хід технологічного процесу може записуватись не лише до власної внутрішньої СУБД, а й до будь-яких інших реляційних СУБД, як комерційних – **MS SQL Server®, MS Access, Oracle®, Sybase,** так і безкоштовних – **Firebird, MySQL**, **Postgre.** Завдяки подібній відкритості, додатки на базі Trace Mode легко інтегруються до корпоративних інформаційних систем підприємств.

Для інтеграції з реляційними СУБД та іншими додатками в Trace Mode вбудовано спеціальний модуль – редактор SQL-запитів, що забезпечує повну підтримку мови SQL та забезпечує відкритість взаємодії як

з потужними промисловими реляційними СУБД, так і з популярною СУБД MS Access.

Редактор SQL-запитів Trace Mode дозволяє налаштувати джерело даних ODBC, протестувати зв'язок з БД, переглянути і за необхідністю модифікувати її структуру, підготувати шаблон SQL-запиту і перевірити його працездатність як в режимі емуляції, так і з реальними транзакціями.

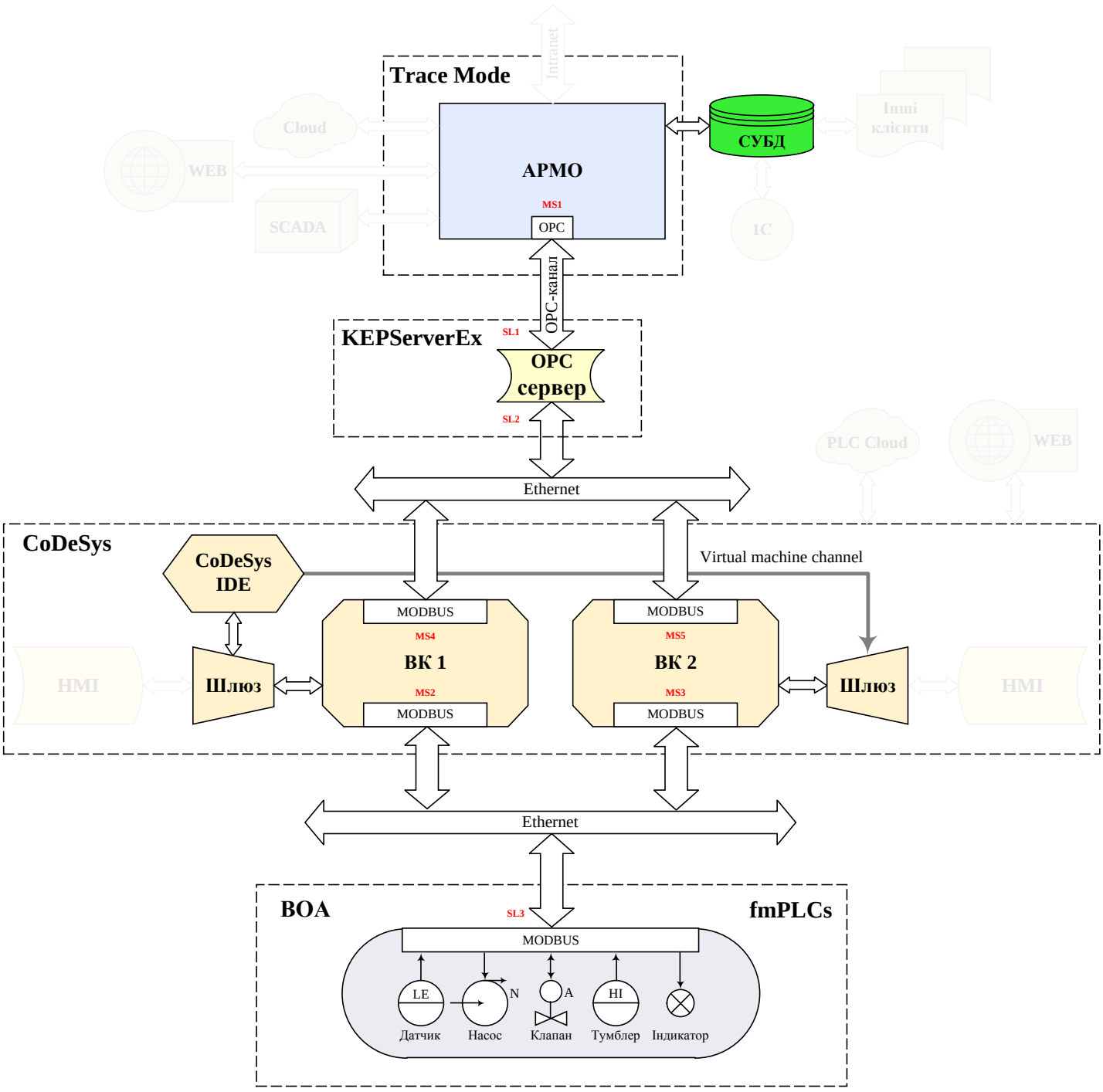


Рис. 3. Створення зв'язку із зовнішньою БД

До засобів редактору SQL-запитів входять:

- редактор структури БД;
- майстер таблиць;
- редактор SQL-запитів;
- майстер SQL-запитів;
- налагоджувач SQL-запитів.

Найбільш зручним засобом для створення шаблону SQL-запиту є спеціальний майстер SQL-запитів (Wizard), що дозволяє користувачу інтерактивно, крок за кроком, програмувати основні операції взаємодії з БД не заглиблюючись в синтаксис мови SQL. Створений за допомогою майстра SQL-запит можна коректувати в текстовому редакторі запитів SQL з підтримкою кольорової підсвітки синтаксису.

Trace Mode не накладає жодних обмежень на SQL-запити, тому за допомогою редактору запитів користувач може реалізувати дуже гнучку взаємодію з зовнішньою БД. Створений запит можна протестувати, якщо відкрито доступ до реальної БД, результати запиту будуть відображені у вікні налагоджувача SQL-запитів.

Для створення нової таблиці БД, в яку передбачається передача даних з Trace Mode, зручно скористатись майстром таблиць. Відредагувати або створити нові поля в існуючих таблицях можна за допомогою редактора структури БД.

10.3 Приклад виконання роботи

Підключення БД

1) У дереві проекту Trace Mode натиснути ПКМ на елементі «**Шаблоны_связей_с_СУБД**» (рис. 3).

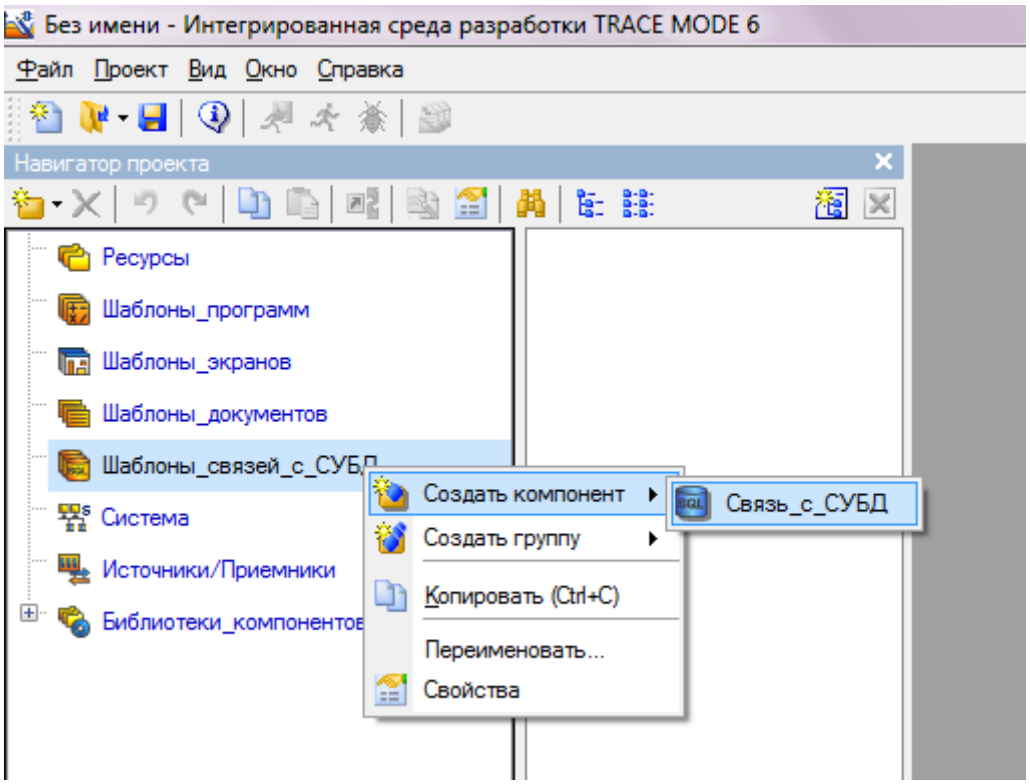


Рис. 3. Створення зв'язку із зовнішньою БД

2) Робимо подвійний клік ПКМ на новоствореному компоненті «**База_даных#1**».

3) Підключення буде розглядатись на базі СУБД **Postgres**. Це повністю безкоштовна потужна СУБД. Усі версії доступні для завантаження на офіційному сайті. Для адміністрування і зручності використання цієї СУБД варто скористатись утилітою **PGAdmin**. Ця утиліта входить в базову комплектацію дистрибутивів для операційної системи Windows. За посиланням http://www.postgresql.org/download/ доступні версії для Windows, Mac OS X, Solaris, Linux, FreeBSD.

В процесі інсталяції СУБД буде створено БД і користувача. У разі виникнення проблем скористайтесь пошуковою системою Google. Після цього запустіть **PGAdmin** і додайте підключення до створеної БД.

На рис. 5 показано як можуть виглядати налаштування для підключення (хост і порт будуть такими самими, решта залежить від користувацького вводу під час установки СУБД).

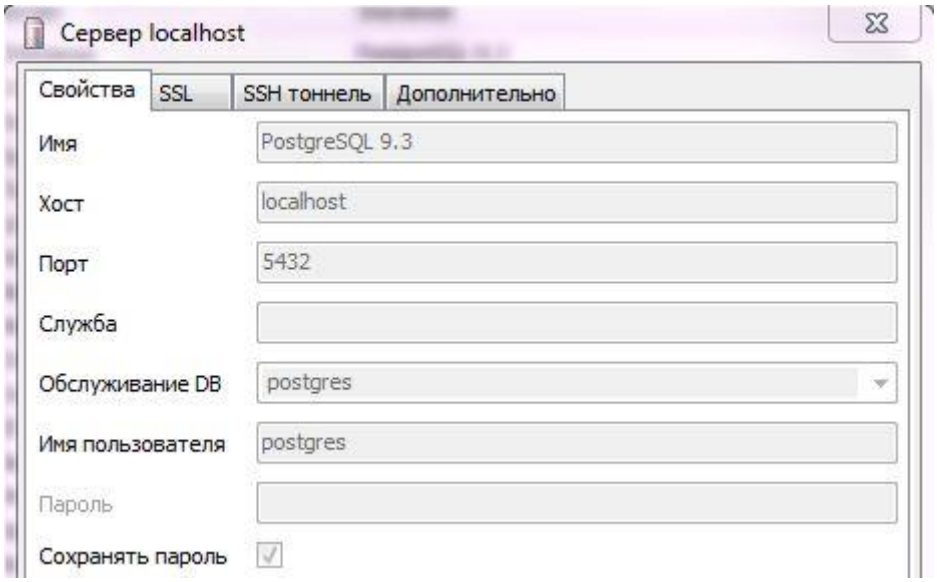


Рис. 5. Дефолтові налаштування підключення СУБД

Скрипт для створення БД за прикладом ОА з попередньої ЛР:

```
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;

GRANT ALL ON SCHEMA public TO public;
GRANT ALL ON SCHEMA public TO postgres;

CREATE TABLE Journal (
  record_id serial PRIMARY KEY,
  le1 real,
  le2 real,
  level_a real,
  level_b real,
  level_c real,
  valve_a real,
  valve_b real,
  valve_c real,
  valve_d real,
  mixer boolean,
  pump real
);
```

На рис. 6 показано встановлене підключення до БД в утиліті **PGAdmin**.

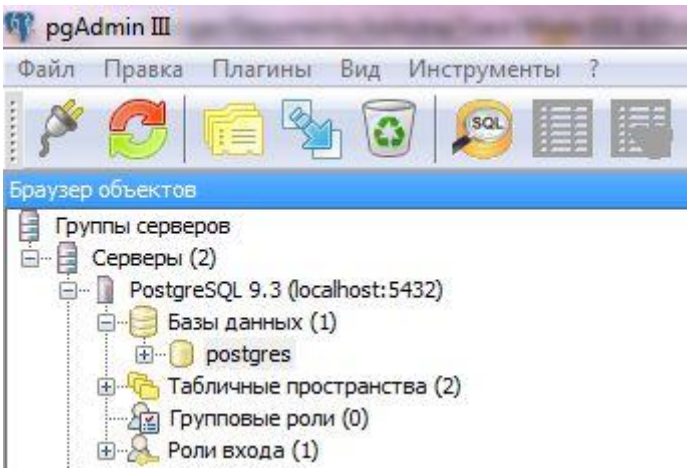


Рис. 6. Перевірка підключення до СУБД утилітою PGAdmin

Після встановлення СУБД необхідно додатково встановити ODBC-драйвер, що доступний для завантаження за посиланням (версія для Windows):

http://ftp.postgresql.org/pub/odbc/versions/msi/psqlodbc_09_03_0100-x64.zip;

http://ftp.postgresql.org/pub/odbc/versions/msi/psqlodbc_09_03_0100.zip.

4) Клікаємо на «**Администратор ODBC**» (рис. 7).

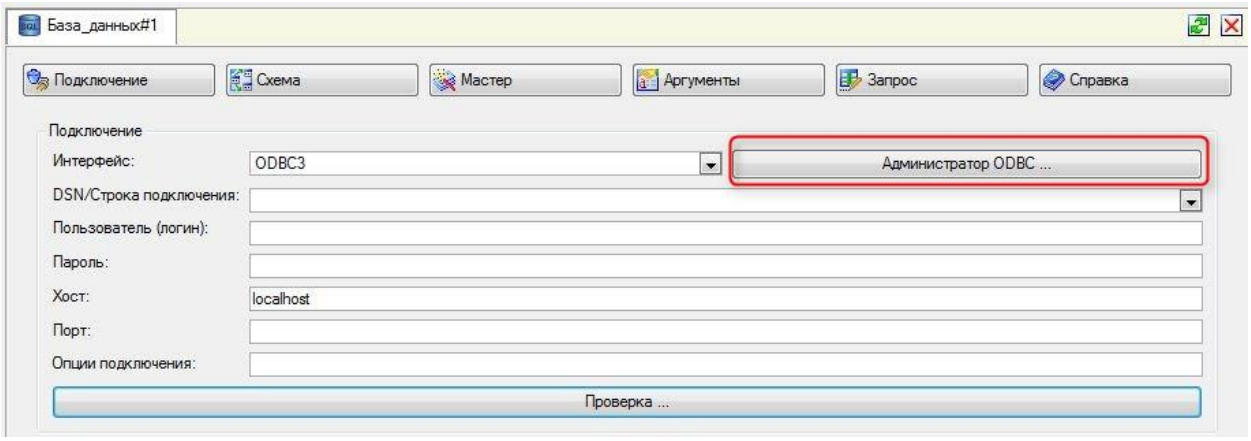


Рис. 7. Підключення в Trace Mode драйверу ODBC

У вікні, що відкрилося клікаємо «**Добавить**» (рис. 8).

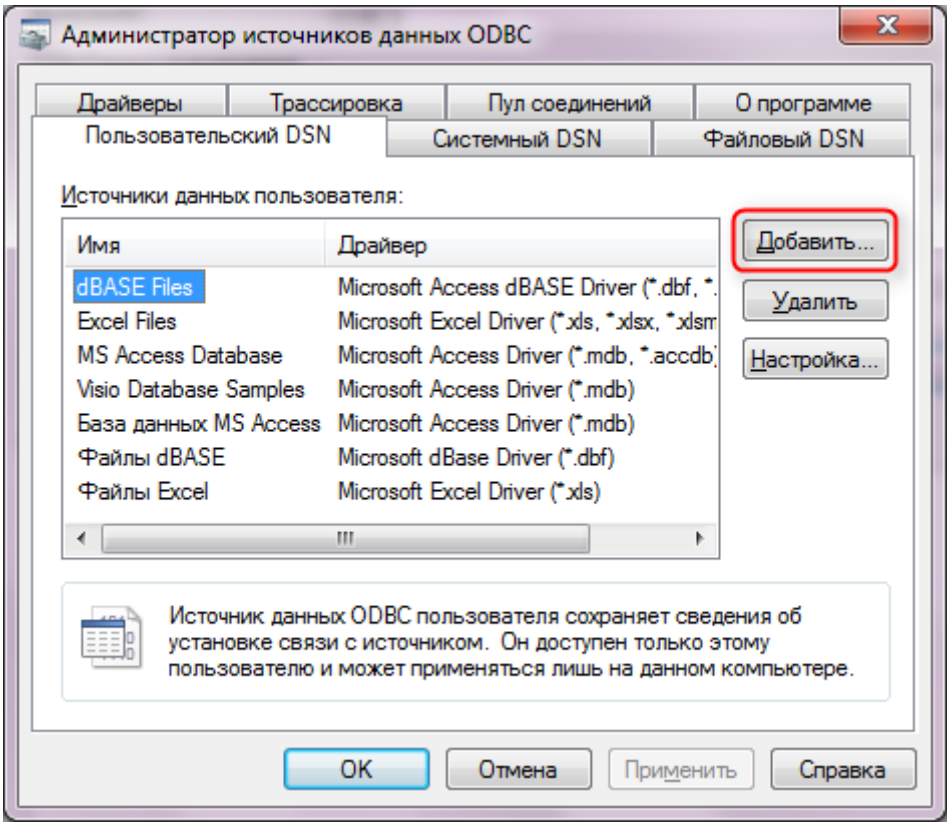


Рис. 8. Додавання dBASE до джерела даних ODBC

Обираємо варіант **PostgreSQL Unicode** (рис. 9).

ЗАУВАЖЕННЯ. У випадку, якщо це підключення не буде працювати, обираємо **PostgreSQL ANSI**. Переключившись з першого на друге, проблеми з відправкою запитів мають вирішитись і можна переключитись назад або продовжити використовувати обране підключення. Ця проблема є проблемою Trace Mode, а не СУБД **PostgreSQL** або використовуваного ODBC-драйвера. У випадку відмови Trace Mode виконувати запити варто перезапустити програму.

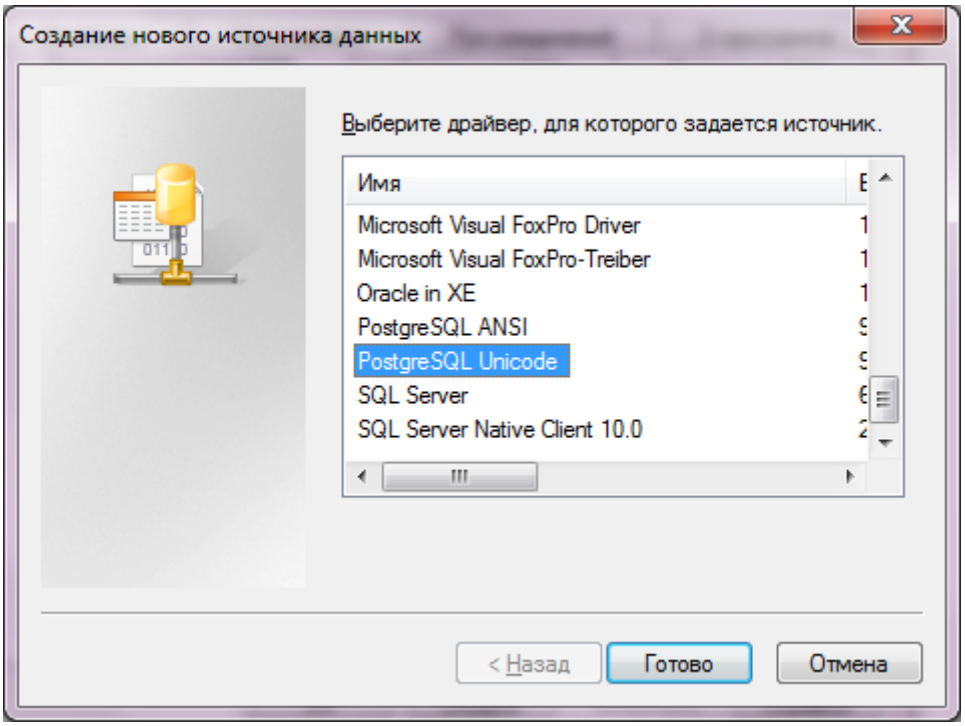


Рис. 9. Вибір зі списку СУБД PostgreSQL

Заповнюємо дані підключення за прикладом як на рис. 10.

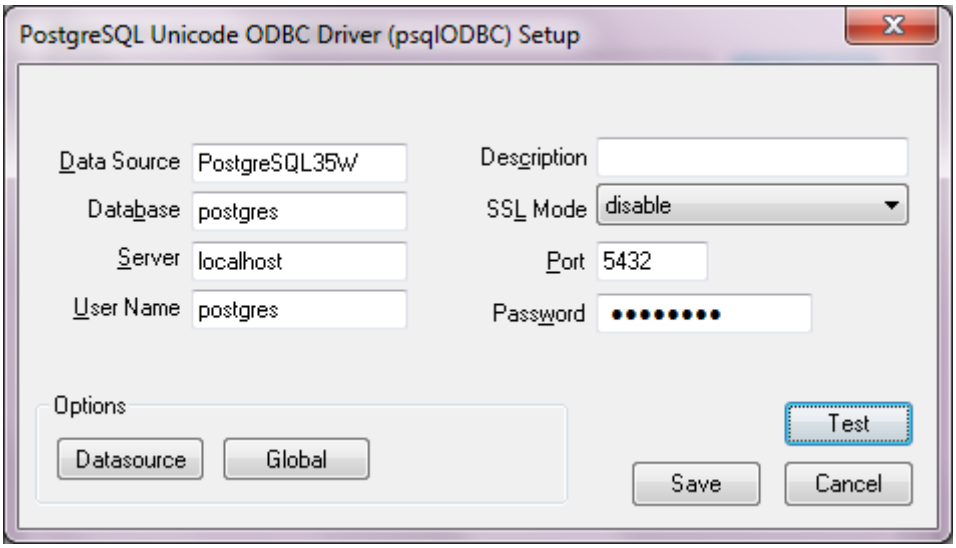


Рис. 10. Налаштування підключення до СУБД

Клікаємо «**Test**» й отримуємо повідомлення «**Connection successful**».

Клікаємо «**Save**».

5) В пункті «**DSN/Строка подключения**» обираємо нашу СУБД **PostgreSQL35W** (рис. 12).

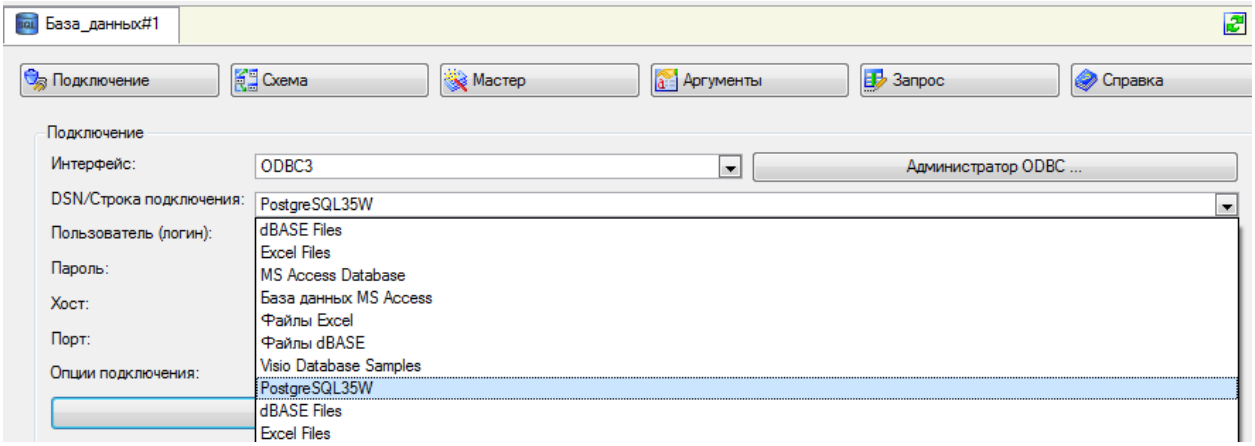


Рис. 12. Вибір та підключення зовнішньої СУБД

Клікаємо «**Проверка**» і отримуємо наступний результат (рис. 13).

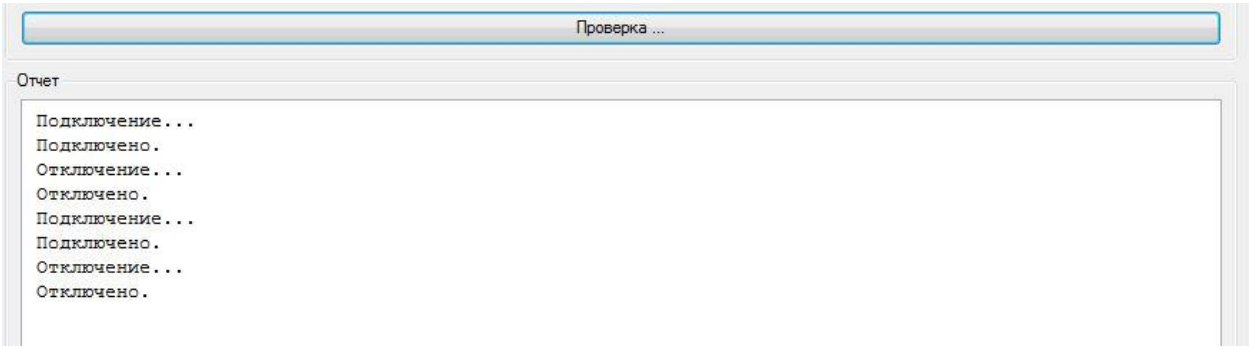


Рис. 13. Перевірка коннекту Trace Mode з БД

10.4 Робота з SQL-запитами

Перейдемо до створення запиту. Це можна зробити двома способами – через вкладки «**Мастер**» та «**Запрос**» (рис. 14).

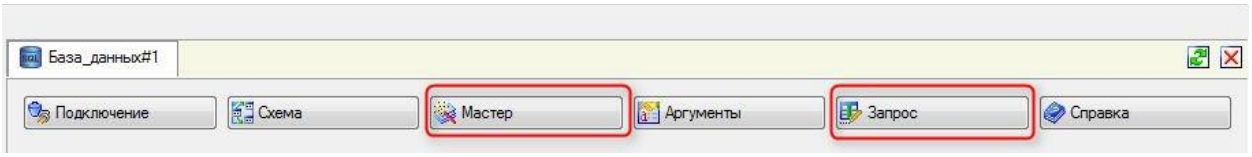


Рис. 14. Створення запиту через вкладку «Мастер»

Використаємо вкладку «**Мастер**» й тип запиту «**SELECT**» (рис. 15).

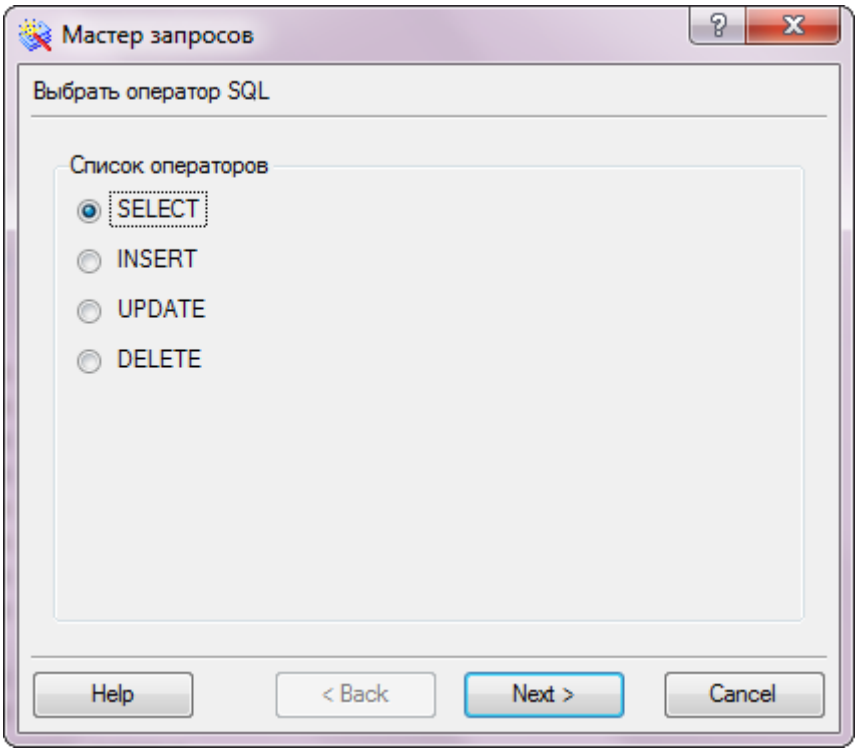


Рис. 15. Вибір запиту до БД типу «SELECT»

Обираємо таблиці, з яких потрібно робити вибірку (рис. 16).

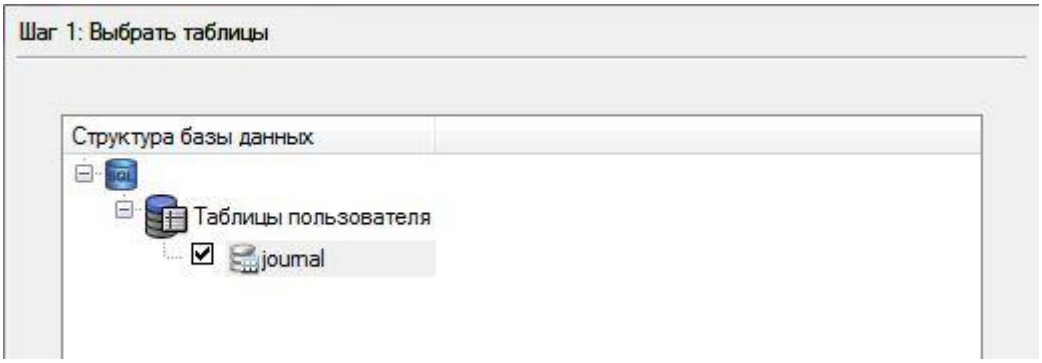


Рис. 16. Вибір таблиці, з якої потрібно робити вибірку

На кроці 2 обираємо необхідні поля з кожної таблиці (рис. 17). У випадку, якщо дані про поля таблиці не підгрузились варто зачекати, перепідключитись до БД, спробувати перезапустити Trace Mode.

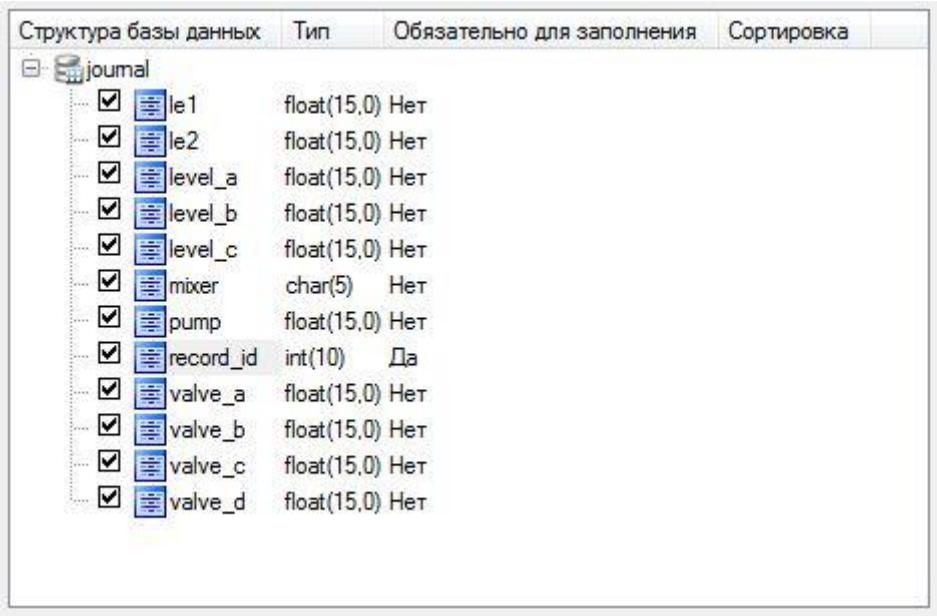


Рис. 17. Вибір необхідних полів (завантажених) з таблиці БД

На кроці 3 встановлюються прив'язки до аргументів запиту (рис. 18).

ЗАУВАЖЕННЯ. Прив'язки необхідно зробити для **УСІХ** полів, що дістаються з БД навіть якщо значення ніколи не буде використовуватись. Якщо кількість аргументів не буде співпадати з кількістю полів, що містить вибірка, Trace Mode не зможе правильно її відображати. Назва аргументу значення не має.

Після цього створюємо прив'язку до поля відповідно до рис. 19. Пункт 1 – додавання нового аргументу. Пункт 2 – встановлення читабельного імені. Пункт 3 – прив'язка. Не забувайте вказати правильний тип даних.

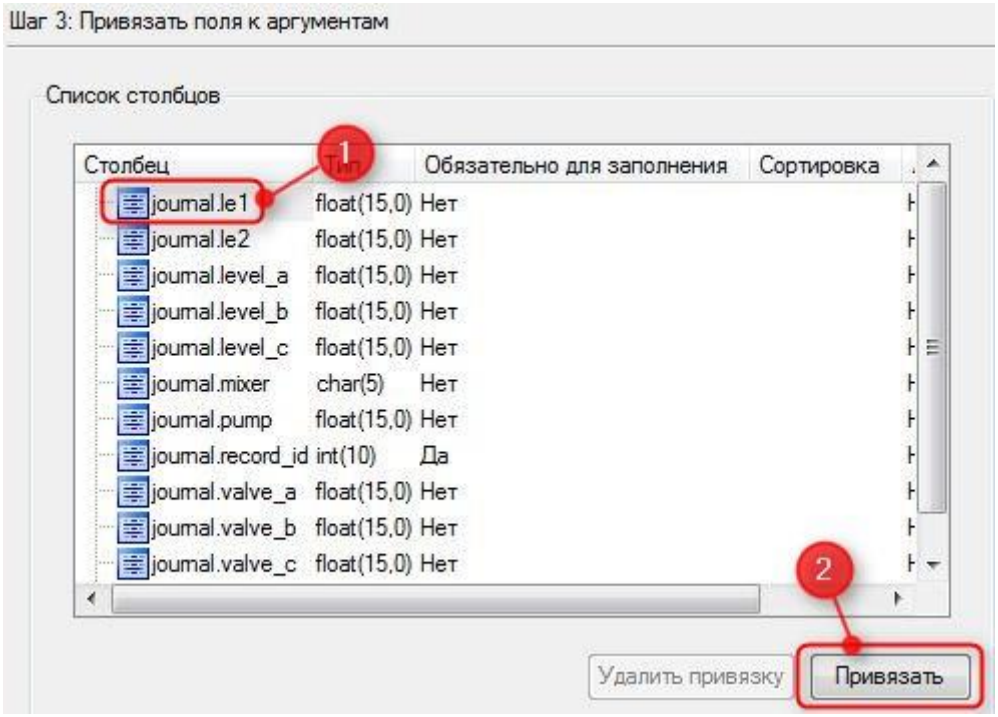


Рис. 18. Встановлення прив'язки до аргументів запиту

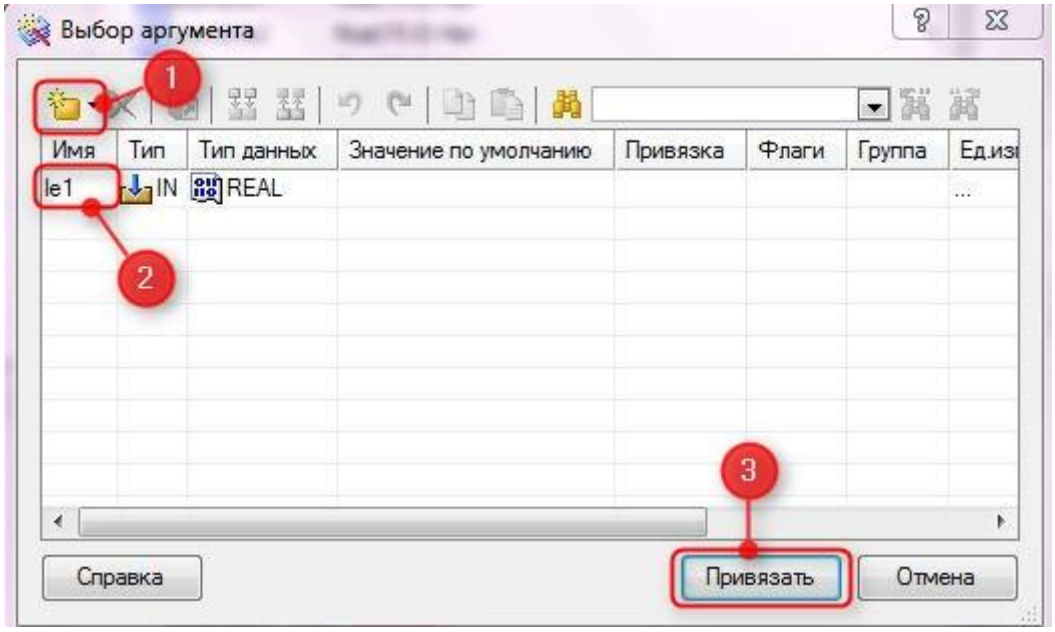


Рис. 19. Створення прив'язки до поля

Фактично ці прив'язки є аліасами і їх варто використовувати не тільки для задавання умов, а й для підвищення читабельності виводу. Без застосування аліасу до поля вивід за цим полем не буде мати жодного зрозумілого підпису. Пам'ятайте, тип даних сигналу mixer – BOOL.

На кроці 4 встановлюються умови (WHERE). Клікаємо «**Добавить**» (рис. 20).

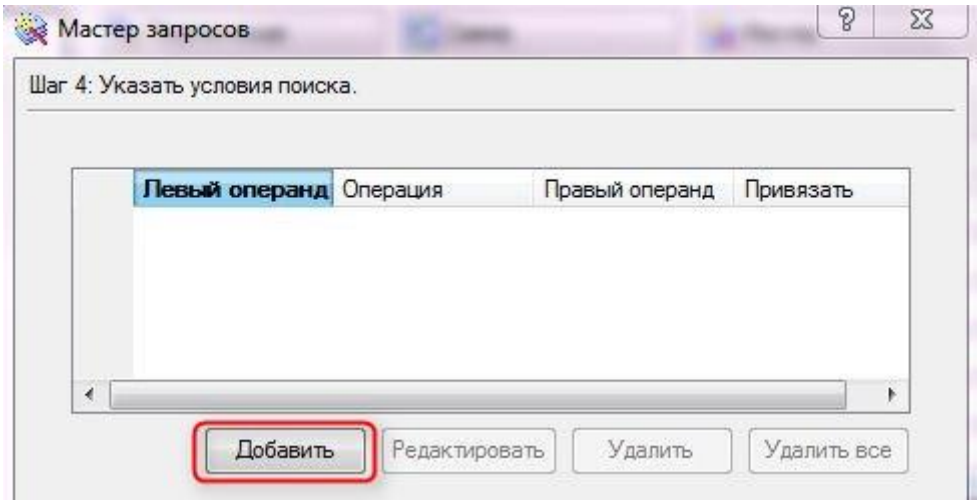


Рис. 20. Ініціація створення запиту

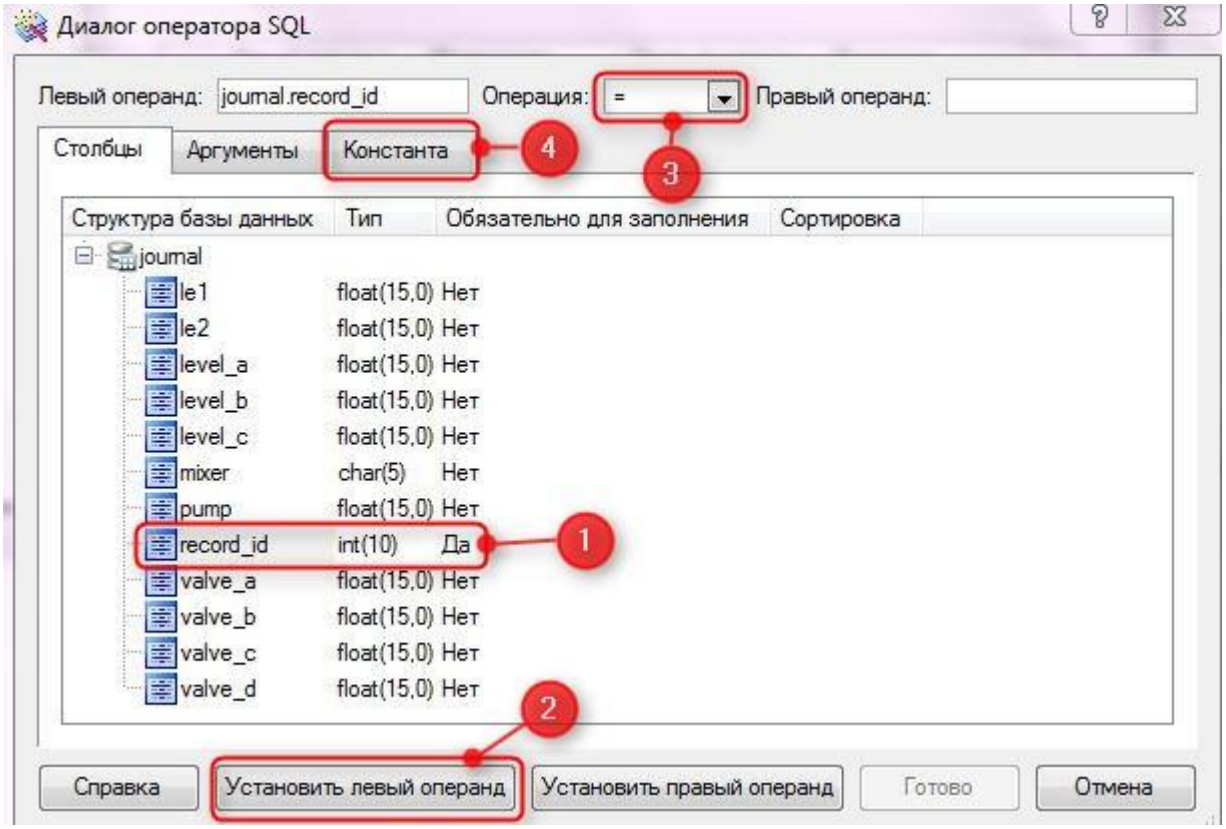


Рис. 21. Формування запиту за операндами

У вікні, що відкрилося (рис. 21) виконуємо наступні дії:

1) Обираємо поле, яке буде лівим операндом виразу.

2) Клікаємо «**Установить левый операнд**».

3) Обираємо необхідну операцію (у даному випадку «=»).

4) Клікаємо на вкладку «**Константа**».

У новій вкладці (рис. 22) обираємо значення константи, встановлюємо правий операнд і клікаємо «**Готово**».

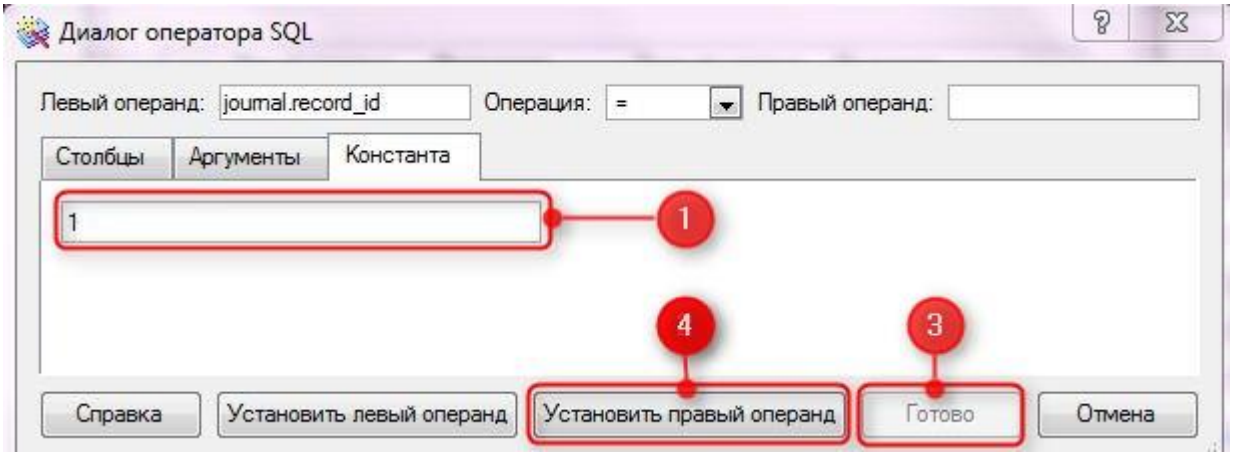


*Рис. 22.*Додавання запиту в Діалозі оператора SQL

Переходимо до вкладки «**Запрос → Выполнить запрос**» (рис. 23).

Рис. 23. Скрит запиту до БД

В такому непривабливому вигляді (рис. 24) подається результат виконання запиту. Можна помітити, що усі поля без аліасів взагалі не мають імені, розділення на рядки можна зробити лише інтуїтивно, хоча виконавши такий самий запит через утиліту **PGAdmin** видно, що БД повертає результат у вигляді набагато більш зрозумілої таблиці.

Будьте пильні, у випадку помилки в запиті ніякого повідомлення про помилку не буде, запит просто не виконається. Тому перевіряйте свої запити за допомогою утиліти **PGAdmin**.

ЗАУВАЖЕННЯ. Наша таблиця в БД на даний момент порожня. Тестові дані (таблиція БД) наведено після рис. 47.

Рис. 24. Результат запиту зчитування з БД

Тепер зайдіть до вкладки «**Запрос**» і видаліть WHERE …. Використаємо цей запит далі, під час виведення даних з БД на Екран АРМО.

10.5 Створення запитів на запис в БД

Створимо запит на запис до БД і новий канал зв'язку з БД (рис. 25). Trace Mode зберігає лише один груповий запит в одному каналі.

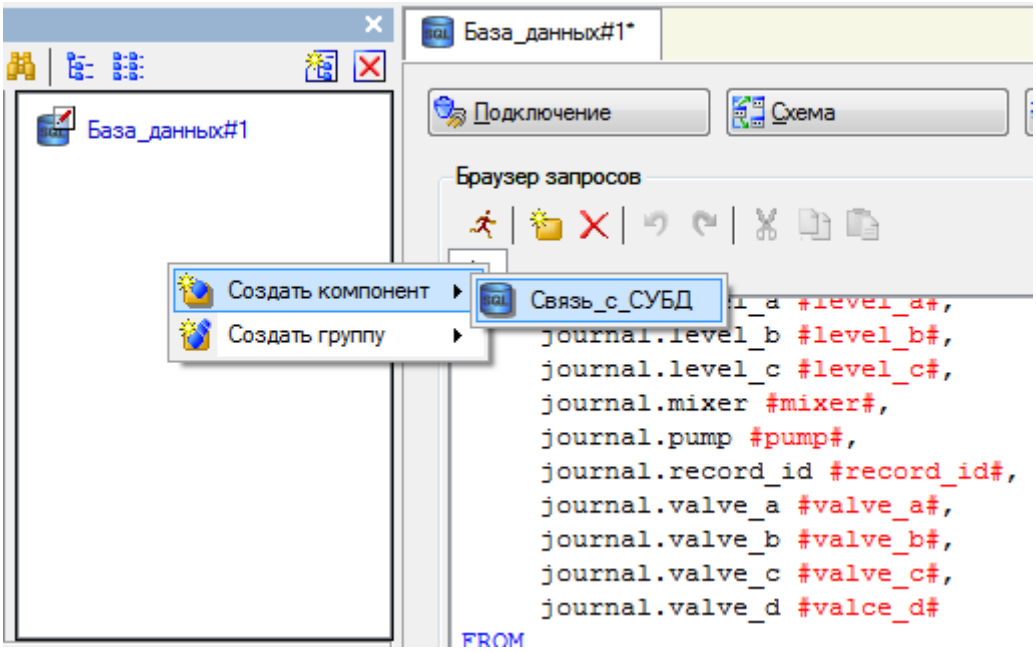


Рис. 25. Створення запиту на запис у БД

Заходимо до вкладки «**Мастер**». Обираємо тип запиту **INSERT** (подібно до рис. 15). На кроці 1 обираємо таблицю, на кроці 2 додаємо необхідні прив'язки, потрібно вказати правильні типи даних, а також тип «**IN/OUT**».

ЗАУВАЖЕННЯ. Прив'язку до record_id створювати не потрібно. У скрипті визначено тип даних «**serial**», це означає, що **Postgres** сам буде заповнювати це поле, відповідно кожного разу інкрементуючи значення.

Приклад правильного результату показано на рис. 26.

```
INSERT INTO journal
(
    le1,
    le2,
    level_a,
    level_b,
    level_c,
    mixer,
    pump,
    valve_a,
    valve_b,
    valve_c,
    valve_d
)
VALUES
(
    '#le1#',
    '#le2#',
    '#level_a#',
    '#level_b#',
    '#level_c#',
    '#mixer#',
    '#pump#',
    '#valve_a#',
    '#valve_b#',
    '#valve_c#',
    '#valve_d#'
)
```

Рис. 26. Приклад запиту на запис у БД

10.6 Інтеграція з СУБД

1) У вузлі «**Система**» створити новий вузол «**RTM_1**».

2) Для виконання запитів в реальному часі потрібно перетягнути увесь вузол «**Шаблоны_связей_с_СУБД**» в «**Система → RTM_1**». В результаті були створені канали виклику шаблонів зв'язку з СУБД (рис. 28).

3) Створити канали, у які будуть записуватись отримані вибірки даних. В групі «**Канали**» новий компонент «**CALL**» (рис. 29).

Робимо подвійний клік ЛКМ на створений CALL. Вибираємо тип виклику «**ChGroupReq**» (рис. 30).

Правий клік на «**CALL → Создать по шаблону → 11**». В результаті мають додатись ще 11 аналогічних каналів. Варто змінити імена каналів на більш змістовні (рис. 31). Ці канали будуть використовуватись для **INSERT**, перший для **SELECT**.

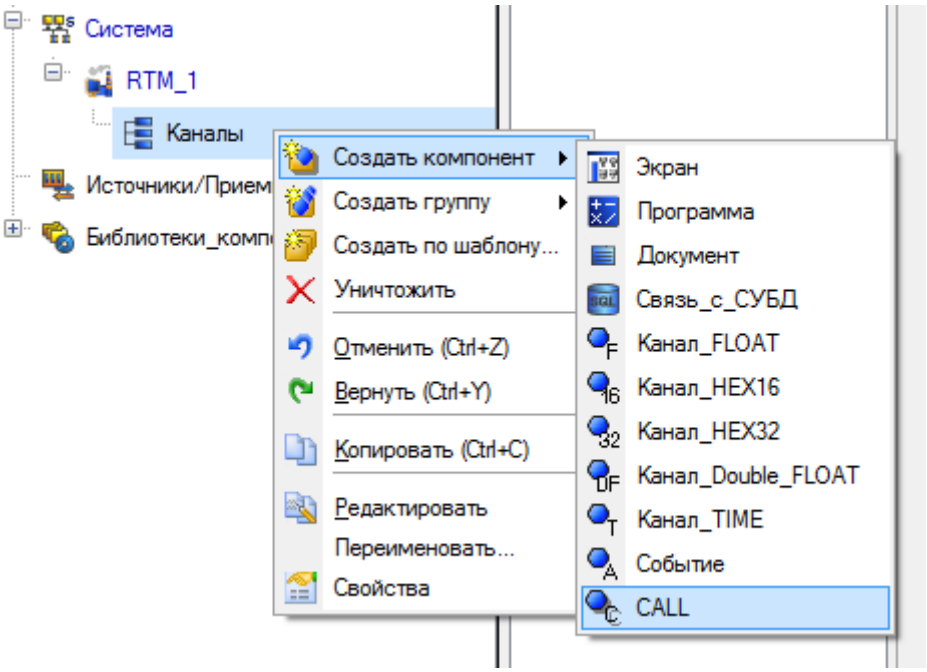


Рис. 29. Створення нового компоненту CALL у групі «Каналы»

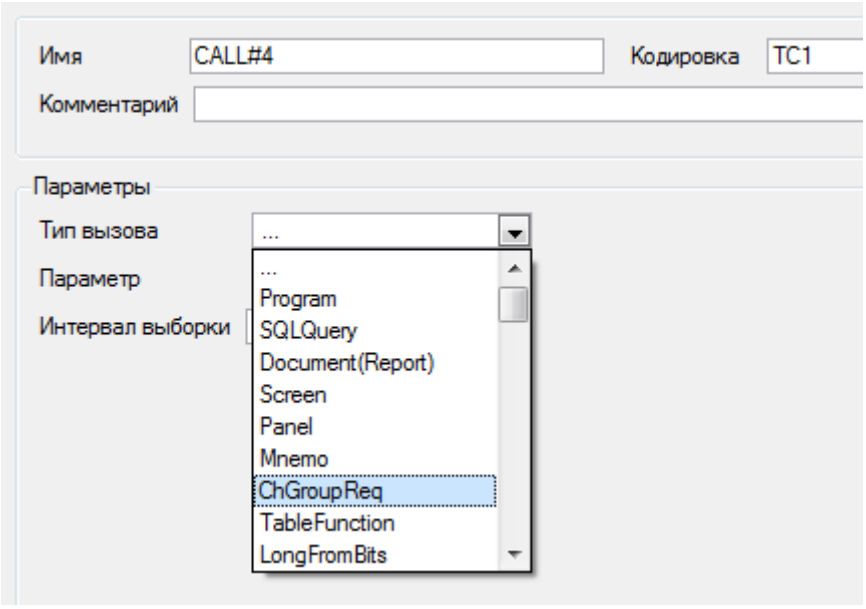


Рис. 30. Вибір типу виклику

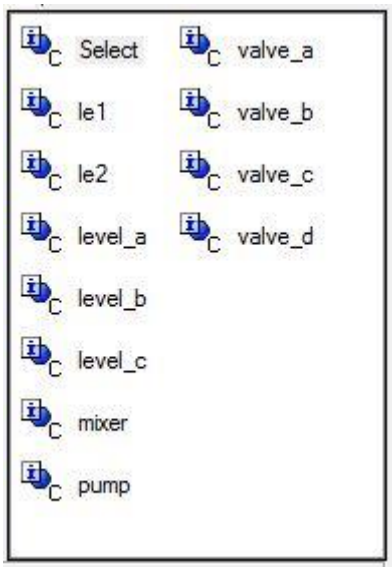


Рис. 31. Створені за шаблоном канали виклику з БД

Повернемось до редактору першого каналу, перейдемо до вкладки аргументи і створимо кілька аргументів (рис. 32).

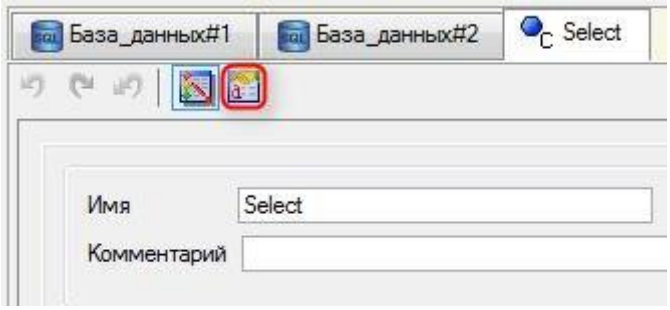


Рис. 32.

Перейдемо до аргументів другого каналу (`LE1`) який відповідає одному з аргументів, що використовувались в **INSERT**. Створимо аргументи відповідно до типу аргументів з пункту 2. В даному випадку це два аргументи типу REAL, також задамо їм значення за замовчуванням (рис. 33).

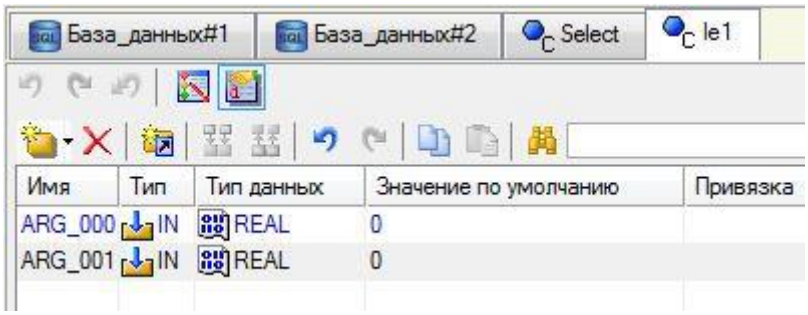


Рис. 33. Створення аргументів відповідно до типів даних REAL

У інших каналах налаштуємо аргументи відповідно до значень аргументів з пункту 10. Тепер потрібно прив'язати створені канали до каналів виклику шаблонів зв'язку з СУБД.

4) Відкриємо друге вікно навігації (рис. 34).

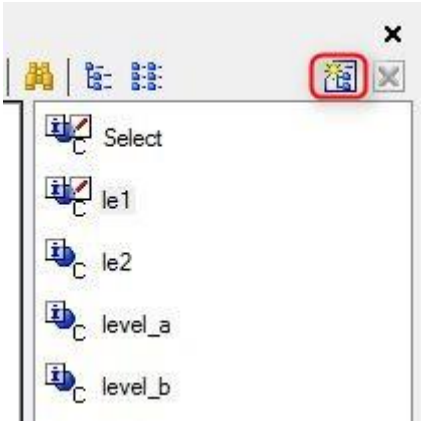


Рис. 34. Вікно навігації каналів

Клікаємо ПКМ на «**База_данных#1:1**» і вибираємо «**Редактировать**» (рис. 35).

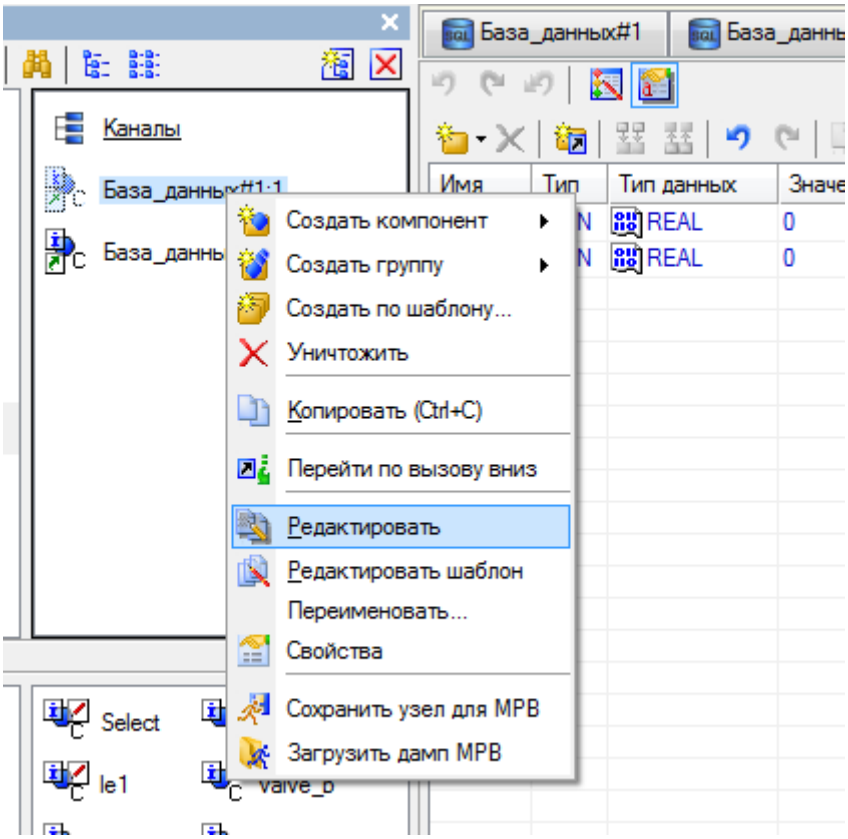


Рис. 35. Редагування каналу з БД

Переходимо до вкладки «**Аргументы**» і перетягуємо канал Select до відповідної комірки «**Привязка**» (рис. 36).

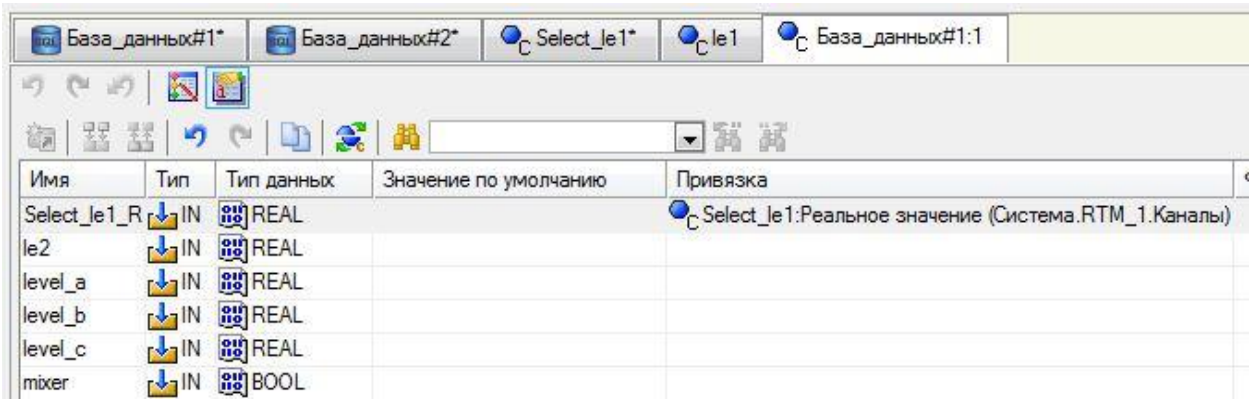


Рис. 36. Прив'язка каналу Select

За аналогією потрібно додати ще відповідну кількість каналів до каналу Select_le1 (Select було перейменовано) для усіх інших аргументів і прив'язати їх.

За аналогією до «**База_данных#1:1**» переходимо до редагування «**База_данных#2:2**». Змінюємо тип на «**Output**», оскільки необхідно виконувати запис до БД (рис. 37).

Рис. 37. Зміна типу каналу на Output

Після цього виконаємо прив'язку відповідних каналів перетягуванням (рис. 38).

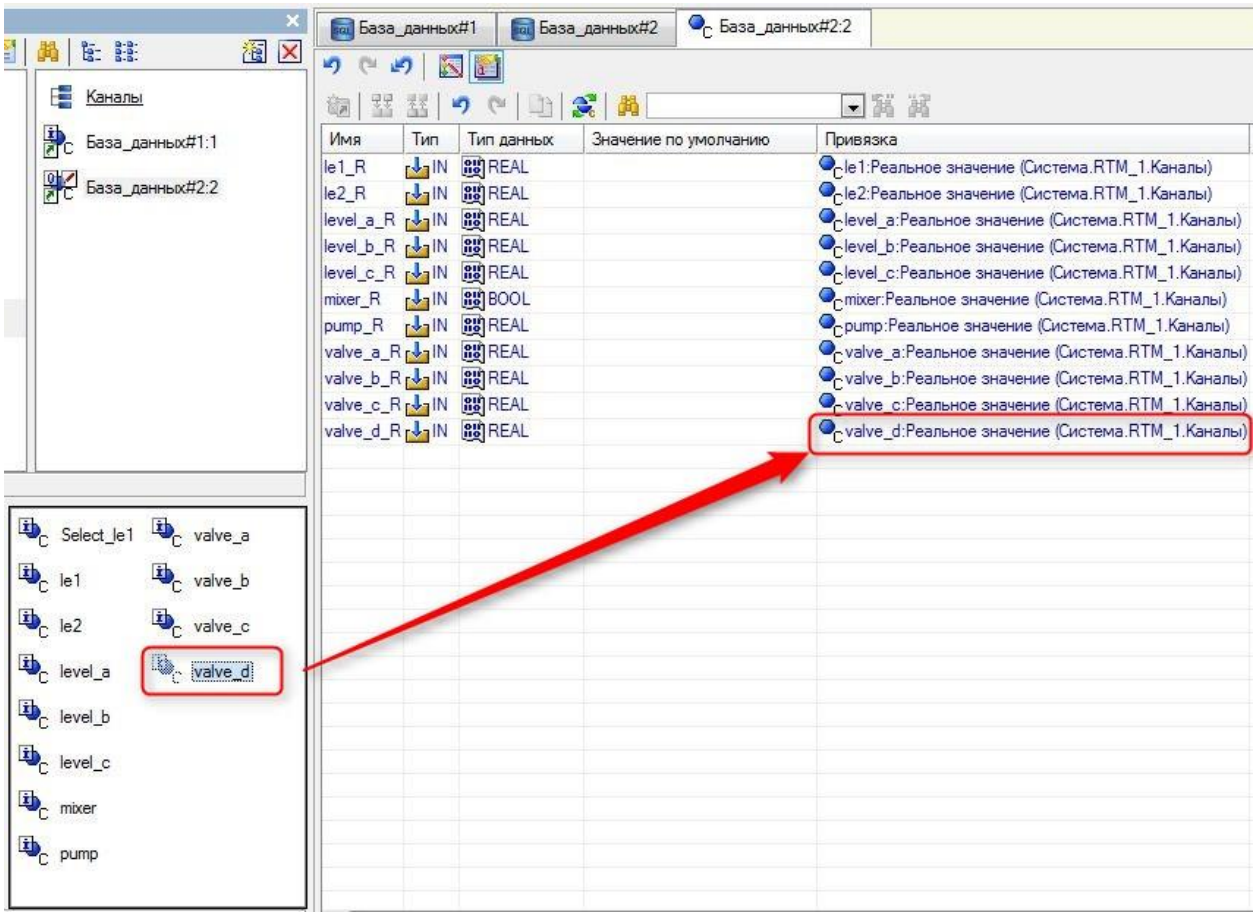


Рис. 38. Прив'язка всіх каналів шляхом перетягування

Збережемо проект, а також збережемо його для МРЧ АРМО (рис. 39).

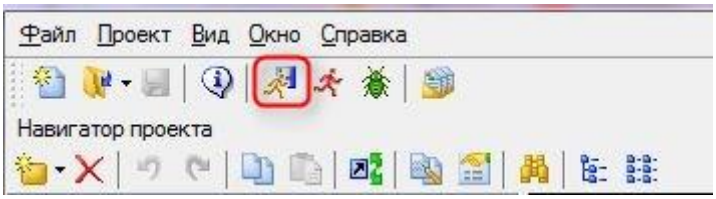


Рис. 39. Збереження проекту в МРЧ

10.7 Виведення інформації з БД до ГЕ «База данных»

Методика роботи з каналами, їх прив'язки до аргументів Екрану детально розглядались в ПрІС-2.ЛР №4. Запис до БД необхідно виконати самим, передаючи відповідні значення в канали, які були прив'язані до запиту **INSERT**.

Необхідно створити новий Екран АРМО. Далі виконати подвійний клік ЛКМ на створеному компоненті Екран й додати ГЕ «**База данных**» (рис. 41, рис. 42).

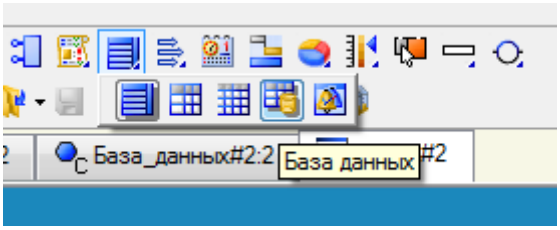


Рис. 41. Додавання ГЕ «База данных» до Екрану

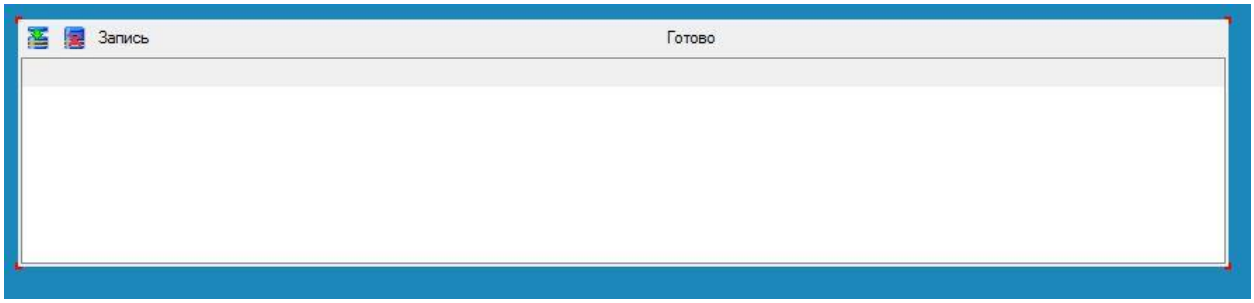


Рис. 42. Приблизний вигляд Екрану показано на рис. 42.

Далі шляхом перетягування необхідно створити новий аргумент Екрану з «**База_данных#1:1**» (рис. 43).

Робимо подвійний клік ЛКМ на ГЕ «**База данных**», відкриються налаштування даного ГЕ.

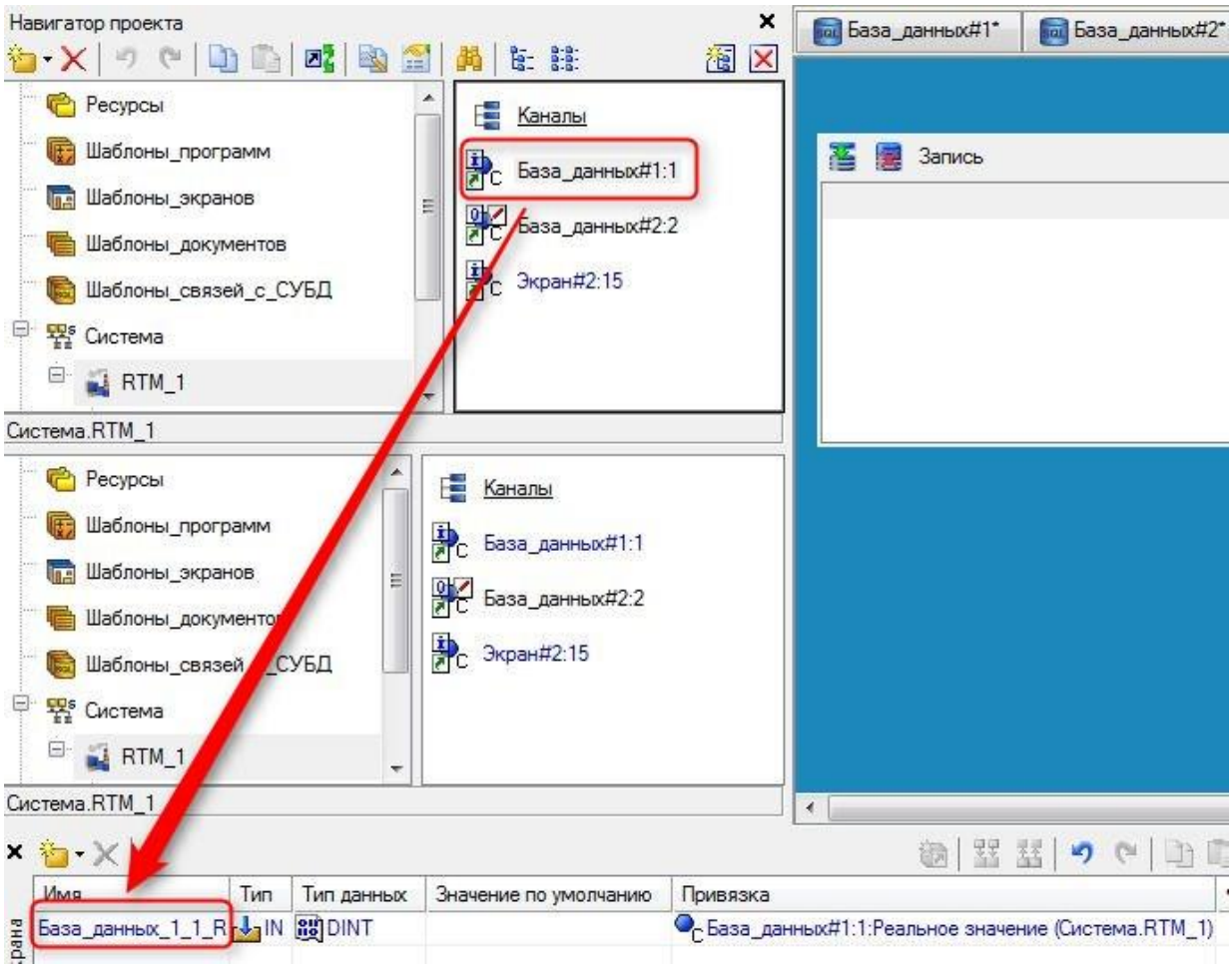


Рис. 43. Створення нового аргументу Екрана

Клікаємо на «**Привязка**» і обираємо доданий аргумент Екрану (рис. 45). Клікаємо «**Готово**».

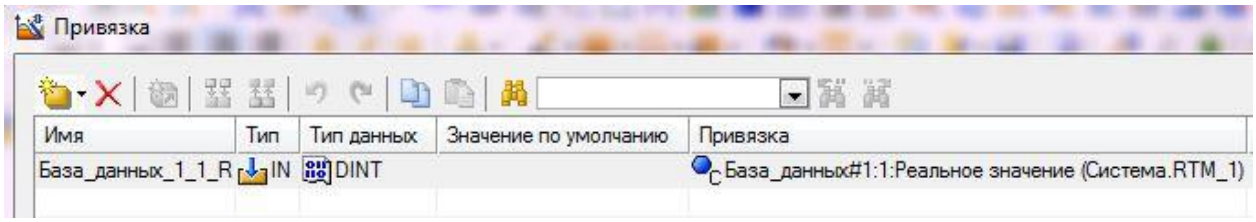


Рис. 45. Прив'язка каналу

Зберігаємо проект, зберігаємо проект для МРЧ АРМО та запускаємо профайлер проекту.

Запускаємо виконання запиту, в ГЕ «**База данных**» натисканням на  і отримуємо результат на рис. 47.

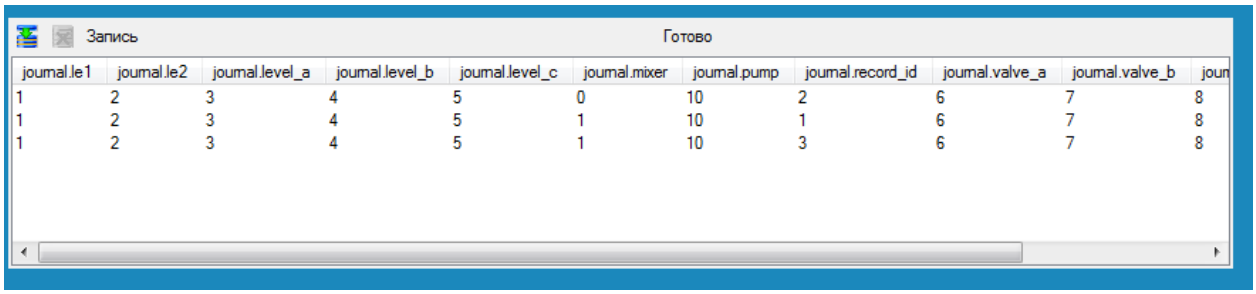


Рис. 47. Результат виконання запиту даних з БД та відображення у ГЕ

Тестові дані були згенеровані за допомогою наступного скрипта:

```
INSERT INTO Journal (le1, le2, level_a, level_b, level_c, valve_a, valve_b, valve_c, valve_d, mixer, pump) VALUES
(1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, true, 10.0),
(1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, false, 10.0),
(1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, true, 10.0);
```

10.8 Контрольні запитання

1) Що таке БД та СУБД?

2) Які особливості встановлення зовнішньої СУБД Postgres та користування утилітою адміністрування PGAdmin?

3) Поясність процедуру створення SQL-запитів в Trace Mode.

4) Як виконати прив'язку полів БД до аргументів запиту?

5) Опишіть послідовність інтеграції СУБД із Trace Mode.

6) Як вивести інформацію з БД до ГЕ Екрану?

10.9 Контрольне завдання

Створити таблицю/і в БД та забезпечити запис/зчитування виробничих параметрів з АРМО в цю БД та навпаки на базі вихідного індивідуального завдання, наведеного у ЛР №1 – файл «**ПрІС-1. ЛР.БІЗ.pdf**».

Компоненти програм починаються ідентифікатором **pip_** (див. п.1.6).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф. В.П. Дьяконова. – М.: СОЛОН–Пресс, 2004. – 256 с.: ил. – (Серия «Библиотека инженера»).
2) Елизаров И.А. Технические средства автоматизации: программирование контроллеров в среде ISaGRAF: лабораторные работы / И.А. Елизаров, А.А. Третьяков, В.Н. Назаров, М.Н. Солуданов. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2008. – 24 с.
3) Елизаров И.А. Технические средства автоматизации. Программно-технические комплексы и контроллеры / И.А. Елизаров, Ю.Ф. Мартемьянов, А.Г. Схиртладзе, С.В. Фролов. – М.: Изд-во Машиностроение-1, 2004. – 180 с.
4) 3S – Smart Software Solutions GmbH. Руководство пользователя: CoDeSys V3, установка и первый запуск. Редакция 3.0. Русская редакция ПК Пролог.
5) Среда программирования CoDeSys и программное обеспечение для ОВЕН ПЛК. http://www.kipservis.ru/pribory_owen/codesys.htm [Електронний ресурс].
6) CoDeSys. http://www.prolog-plc.ru/3s/OEM1.htm [Електронний ресурс].