МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ВОЛОДИМИРА ДАЛЯ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт з дисципліни

«МІКРОКОНТРОЛЕРИ ТА ЇХ ПРОГРАМУВАННЯ »

(*для здобувачів вищої освіти денної та заочної форм навчання спеціальності 151 Автоматизація та комп'ютерно–інтегровані технології*)

(*Електронне видання*)

З А Т В Е Р Д Ж Е Н О
на засіданні кафедри комп'ютерно-інтегрованих систем управління
Протокол № 4 від 02.12.2021 р.

СЄВЄРОДОНЕЦЬК 2021

УДК 681.518

Методичні вказівки до виконання лабораторних робіт з дисципліни «Мікроконтролери та їх програмування» (для здобувачів вищої освіти денної та заочної форм навчання спеціальності 151 Автоматизація та комп'ютерно–інтегровані технології) / Укладачі: О.І. Проказа, О.В. Кузнецова. – Сєвєродонецьк: Вид-во СНУ ім. В. Даля, 2021. - 128 с.

Наведені методичні вказівки до виконання лабораторних робіт з дисципліни «Мікроконтролери та їх програмування» для здобувачів вищої освіти денної та заочної форм навчання. Описані методи і принципи розробки управляючих програм для мікроконтролерів в середовищі CoDeSys, постановка завдань, варіанти завдань, приклади виконання.

Укладачі: О.І. Проказа, к.т.н., доц.

О.В. Кузнецова, ст. викл.

Рецензент М.Г. Лорія, д.т.н., проф.

ЗМІСТ

ВСТУП

Методичні вказівки складені відповідно до робочої програми з курсу «Мікроконтролери та їх програмування» спеціальності 151 – Автоматизація та комп’ютерно–інтегровані технології.

«Мікроконтролери та їх програмування» – це технічна дисципліна, в якій вивчають архітектуру мікроконтролерів, інструменти програмування контролерів та мови стандарту МЕК.

Мета дисципліни «Мікроконтролери та їх програмування» – ознайомлення студентів з основними технічними засобами на основі мікроконтролерної та мікропроцесорної техніки, з сучасними засобами їх програмування.

Завдання дисципліни «Мікроконтролери та їх програмування» – закріплення практичних знань та умінь студентів застосовувати системи програмування мікроконтролерів для вирішення конкретних технологічних задач.

В результаті вивчення дисципліни студент зобов’язаний

знати:

- загальні принципи побудови мікроконтролерів;
- архітектуру та склад типових сімейств мікроконтролерів;
- технічні особливості використання мікроконтролерів різних фірм;
- стандарт МЕК 61131 в розділі програмування мікроконтролерів;
- апаратні та інструментальні засоби програмування мікроконтролерів;

вміти:

- вибирати мікроконтролери за вимогами, що пред'являються до автоматизованого технологічного процесу;
- визначати структуру і вибирати засоби сполучення контролера з вимірювальними датчиками та виконавчими механізмами;
- розробляти проєкти систем контролю і управління;
- застосовувати сучасні інструментальні системи програмування мікроконтролерів.

ЗАГАЛЬНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Порядок виконання лабораторних робіт

Для виконання лабораторних робіт з курсу «Мікроконтролери та їх програмування» необхідно уважно ознайомитися з теоретичними основами кожної роботи, приведеними в методичних вказівках. Для більш глибокого ознайомлення з матеріалом, необхідно звертатися до конспекту лекцій, а також до технічної літератури, на яку приводяться посилання у вказівках.

Якщо під час вивчення дисципліни з'являться запитання, на які не знаходиться відповідь у конспекті лекцій або технічній літературі, необхідно звертатись за консультацією до ведучого лектора.

Лабораторна робота складається з:

- одержання індивідуального завдання;
- одержання шуканих результатів;
- оформлення й захист звіту з лабораторної роботи.

Лабораторні роботи необхідно виконувати за допомогою персонального комп'ютера або ноутбука з використанням системи CoDeSys.

Правила оформлення звітів з лабораторних робіт

Звіт повинен містити:

- титульний аркуш із найменуванням лабораторної роботи й даними виконавця;
- мета роботи;
- постановка задачі відповідно до варіанту;
- порядок і результати дослідження в числовому і графічному вигляді.

аналіз результатів і висновки.

ЛАБОРАТОРНА РОБОТА № 1

Тема: ВИВЧЕННЯ СЕРЕДОВИЩА CODESYS РОЗРОБКИ ПРОГРАМ ДЛЯ КОНТРОЛЕРІВ

Мета роботи: ознайомлення з середовищем CoDeSys, освоєння інтерфейсу й приймань роботи з інструментом програмування контролерів CoDeSys.

Постановка задачі: ознайомлення з демонстраційними проєктами середовища CoDeSys

Короткі теоретичні відомості

Програмовані логічні контролери

Програмовані логічні контролери, скорочено ПЛК (PLC – Programmable Logic Controller) вперше з'являються у 1969р. для автоматизації автомобільної промисловості. Зараз ПЛК використовуються в енергетиці, хімічній промисловості, системах забезпечення безпеки, харчовому виробництві, машинобудуванні, транспорті тощо. Типовий ПЛК являє собою мікропроцесорний блок з деякою кількістю входів і виходів для підключення датчиків та виконавчих механізмів (рисунок 1). Входи і виходи зазвичай роблять стандартними, тому при зміні алгоритму роботи не потрібно ніякої зміни апаратної частини.

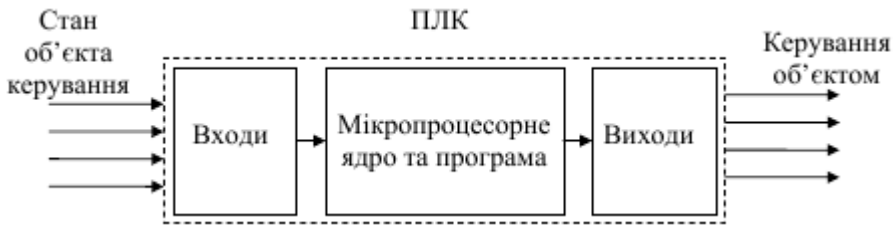

Рисунок 1 – Принцип роботи ПЛК

Програмований контролер – це програмно керований дискретний автомат, що має певну кількість входів, підключених за допомогою датчики до об'єкта керування, і деяку кількість виходів, підключених до виконуючих пристроїв. ПЛК

контролює стан входів і виробляє певні послідовності програмно заданих дій, що відображаються в зміні виходів.

Завданням прикладного програмування ПЛК є реалізація алгоритму керування конкретним об’єктом керування (машиною, агрегатом). Опитування входів і виходів контролер здійснює автоматично, незалежно від способу фізичного з'єднання. Цю роботу виконує системне програмне забезпечення. В ідеальному випадку прикладний програміст зовсім не цікавиться, як приєднані й де розміщені датчики й виконавчі механізми. Крім того, його робота не залежить від того, з яким контролером й якою фірмою він працює. Завдяки стандартизації мов програмування прикладна програма виявляється переносимою. Це означає, що її можна використати в будь-якому ПЛК, що підтримує даний стандарт.

ПЛК призначений для роботи в режимі реального часу в умовах промислового середовища і повинен бути доступним для програмування неспеціалістом в області інформатики.

З самого початку ПЛК призначалися для керування послідовними логічними процесами, що й обумовило слово «логічний» у назві ПЛК. Сучасні ПЛК крім простих логічних операцій здатні виконувати цифрову обробку сигналів, керування приводами, регулювання, функції операторського керування тощо.

Задачі керування вимагають неперервного циклічного контролю. У будь-яких цифрових пристроях неперервність досягається за рахунок застосування дискретних алгоритмів, що повторюються через досить малі проміжки часу. Таким чином, обчислення в ПЛК завжди повторюються циклічно. Одна ітерація, що включає вимірювання і розрахунок впливу, називається *робочим циклом ПЛК*. Виконувані дії залежать від значення входів контролера, попереднього стану і визначаються користувацькою програмою.

При вмиканні живлення ПЛК виконує самотестування і настроювання апаратних ресурсів, очищення оперативної пам'яті даних (ОЗП), контроль цілісності прикладної програми користувача. Якщо прикладна програма

збережена в пам'яті, ПЛК переходить до основної роботи, що складається з постійного повторення послідовності дій, що входять до робочого циклу.

Робочий цикл ПЛК складається з декількох фаз:

1. Початок циклу.
2. Читання стану входів.
3. Виконання коду програми користувача.
4. Запис стану виходів.
5. Обслуговування апаратних ресурсів ПЛК.
6. Монітор системи виконання (системне програмне забезпечення ПЛК).
7. Контроль часу циклу.
8. Перехід на початок циклу.

Стандарт МЕК 61131

Стандарт МЕК 61131 було впроваджено у 1982 р з метою вирішення проблем уніфікації устаткування промислової автоматизації. Він охоплює вимоги до апаратних засобів, монтажу, тестування, документації, зв’язку і програмування ПЛК. Мови програмування відносяться до розділу МЕК 61131-3.

Стандартом описується 5 мов програмування:

– Instruction List (IL) – список інструкцій, асемблероподібна мова;

– Structured Text (ST) – структурований текст, високорівнева мова;

– Sequential Function Chart (SFC) – послідовні функціональні діаграми;

– Function Block Diagram (FBD) – функціонально-блокові діаграми;

– Ladder Diagram (LD) – сходинкова діаграма, мова релейно-контактних схем.

Мови, що увійшли в стандарт, створені на основі найбільш популярних мов програмування, найпоширеніших у світі контролерів. Якщо взяти будь-який контролер, що працює в сучасному виробництві, то його програму можна перенести в середовище МЕК 61131-3 із мінімальними затратами.

Також підтримується мова Continuous Function Chart (CFC), що відрізняється від стандартного FBD можливістю вільного розміщення елементів, створення зворотних зв’язків та вибору порядку виконання блоків.

Комплекс CoDeSys

CoDeSys – це сучасний інструмент для програмування контролерів (CoDeSys утворюється від слів Controllers Development System). Система CoDeSys була виготовлена компанією 3S-Smart Software Solutions GmbH.

CoDeSys надає програмісту зручне середовище для програмування контролерів мовами стандарту МЕК 61131-3. Використовувані редактори та налагоджувальні засоби базуються на широко відомих і добре зарекомендованих принципах, знайомих по інших популярних середовищах професійного програмування (Visual C++ тощо).

Комплекс CoDeSys не прив'язаний до конкретної апаратної платформи, існує кілька модифікацій спеціально оптимізованих під різні мікропроцесори. Для прив'язки до конкретного ПЛК потрібна адаптація програми до низкорівневих ресурсів – розподілу пам'яті, інтерфейсів зв'язку та інтерфейсів уводу-виводу.

Серед особливостей середовища CoDeSys можна відзначити такі:

– пряма генерація машинного коду, що забезпечує високу швидкодію програм користувача;

– повноцінна реалізація мов МЕК;

– «розумні» редактори мов побудовані в такий спосіб, що не дають робити типові для початківців помилки;

– вбудований емулятор контролера, що дозволяє проводити налагодження проєкту без додаткових апаратних засобів;

– вбудовані елементи візуалізації дають можливість створити модель об'єкту керування та проводити налагодження проєкту без виготовлення засобів імітації;

– набір бібліотек і готових сервісних функцій.

Базовий склад комплексу програмування ПЛК складається із двох обов'язкових частин: системи виконання й робочого місця програміста. Система виконання функціонує в контролері. Крім безпосереднього виконання керуючої програми вона забезпечує завантаження коду прикладної програми і функції її налагодження. Система виконання повинна мати зв'язок з комп'ютером робочого місця програміста. Не важливо як фізично організований зв'язок ПК і ПЛК, у найпростішому випадку ПЛК підключається до комп'ютера через стандартний COM-порт (RS-232) нуль-модемним кабелем. В умовах цеху може використовуватися більш завадостійкий і далекобійний інтерфейс (RS-422, RS-485 або струмова петля).

У комплексі CoDeSys посередником між середовищем розробки та ПЛК слугує спеціальний додаток – шлюз зв'язку (gateway). Шлюз зв'язку взаємодіє з інтегрованим середовищем через Windows-сокет з'єднання, побудоване на основі протоколу TCP/IP. Таке з'єднання забезпечує однакову взаємодію додатків, що працюють на одному комп'ютері або в мережі (рисунок 2). Завдяки цьому програміст може абсолютно повноцінно працювати на віддаленому комп'ютері. Причому віддаленість не обмежується рамками локальної мережі. ПК, що виконує роль шлюзу зв'язку, може одночасно взаємодіяти із ПК програміста через Інтернет і із ПЛК через модемне з'єднання.

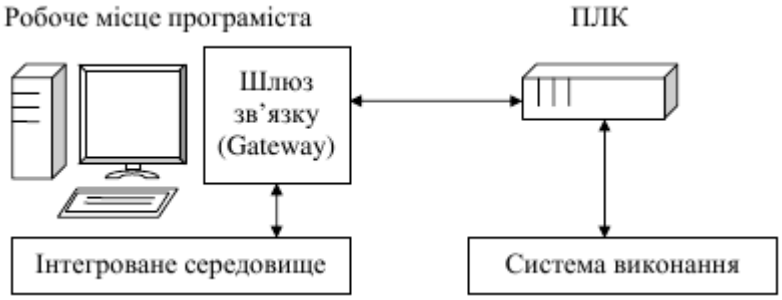

Рисунок 2 – Взаємозв’язок компонентів комплексу CoDeSys

За замовчуванням шлюз зв'язку настроєний на локальну роботу й запускається автоматично при встановленні зв'язку з ПЛК із інтегрованого середовища. Для з'єднання з ПЛК через COM-порт достатньо настроїти параметри драйвера інтерфейсу відповідно до керівництва по застосуванню ПЛК (порт, швидкість, контроль паритету та число стоп-бітів).

Організація роботи у середовищі розробки CoDeSys

Програми у середовищі розробки CoDeSys зберігаються у виді ***проєктів***.

Проект містить ряд різнорідних ***об'єктів***: програмні компоненти (POU - Program Organization Unit), дані різних типів, елементи візуалізації й ресурси. Кожний проект зберігається в окремому файлі.

До ***програмних компонентів*** (**POU**) відносяться функціональні блоки, функції й програми. Окремі POU можуть включати дії (підпрограми). Перший програмний компонент розміщується в новому проекті автоматично і дістає назву **PLC_PRG**. Саме з нього і починається виконання процесу (за аналогією з функцією **main** у мові **C**), з нього будуть викликатися інші програмні компоненти (програми, функції й функціональні блоки). POU можуть викликати інші POU, але рекурсії неприпустимі.

Кожен програмний компонент складається з розділу оголошень і коду. Для написання всього коду POU використовується тільки одна з мов МЕК програмування (IL, ST, FBD, SFC, LD або CFC) (Рисунок 4).

CoDeSys підтримує всі описані стандартом МЕК компоненти. Для їхнього використання досить включити у свій проект бібліотеку **standard.lib**.

Організатор об'єктів (рисунок 3) керує списком усіх об'єктів проекту (рисунок 4). Він завжди знаходиться в лівій частині Головного вікна CoDeSys. У нижній частині організатора об'єктів знаходяться вкладки POU, Типи даних, Візуалізації і Ресурси. Переключатися між відповідними об'єктами можна за допомогою миші або клавіш переміщення.

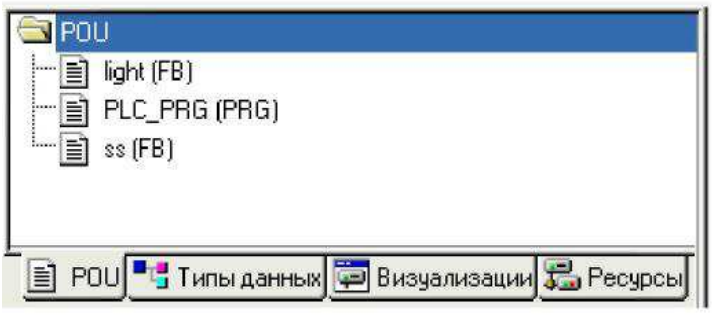

Рисунок 3 – Вікно організатора проекту

Рисунок 4 – Вікна об'єктів проекту

Порядок виконання роботи

1. Запустити середовище розробки CoDeSys. Якщо в організаторі об’єктів автоматично відкривається проект, потрібно закрити його за допомогою меню **Файл → Закрити**.

2. Відкрити демонстраційний проєкт під назвою **Example.pro** за допомогою кнопки на панелі інструментів або меню **Файл→Відкрити**.

3. Занести до звіту перелік програм (**PRG**), функцій (**FUN**) та функціональних блоків (**FB**), які входять до даного проєкту. Цей перелік відображається в організаторі об’єктів. Вибрати програмний компонент **LD_Example** і занести його до звіту.

4. Перейти на вкладку **Візуалізації** у організаторі об’єктів, дослідити які елементи візуалізації входять у даний проект, зовнішній вигляд **LD_Visu** занести в звіт.

5. Увімкнути режим емуляції за допомогою меню **Онлайн → Режим емуляції**.

6. Під’єднати середовище до емуляції ПЛК за допомогою меню **Онлайн → Підключення**. Зверніть увагу на те, що під час під’єднання виконується компіляція проєкту.

7. Запустити проект на виконання за допомогою меню **Онлайн → Старт**.

8. За допомогою кнопок на **S1-S15 LD_Visu** ввімкніть дві лампи. Користуючись схемою POU **LD_Example** вкажіть яку мінімальну кількість кнопок і які саме необхідно ввімкнути, щоб спрацювали лампи.

9. Під час функціонування проєкту дослідіть процес відображення змінних в усіх блоках POU **LD_Example**, занести в звіт.

10. Перейдіть до візуалізації **Polygon**, дізнайтесь для чого необхідні кнопки на цій панелі, занести в звіт

11. Перейдіть до візуалізації **ST_Visu**, натисніть кнопку **Старт** і дослідіть функціонування цього блока, різні положення занести в звіт.

Контрольні питання

1. Що таке ПЛК?
2. Яке призначення ПЛК?
4. Які операції може виконувати ПЛК?
5. З яких операцій складається робочий цикл ПЛК?
6. Які мови програмування описує стандарт МЕК 61131?
7. Які особливості середовища CoDeSys?
8. Що входить до базового складу комплексу CoDeSys?
9. Що містіть проект CoDeSys?
10. Що таке POU?
11. Що містить організатор об’єктів?

ЛАБОРАТОРНА РОБОТА № 2

Тема: РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ПЕРЕМИКАЧАМИ

Мета роботи: отримати практичні навички проєктування в системі CoDeSys. Освоїти емуляцію і візуалізацію виконання проєкту.

Постановка задачі. Створити ПЛК-програму для керування двома перемикачами.

Короткі теоретичні відомості

Проєктування в системі CoDeSys

Для розробки проєкту в CoDeSys необхідно виконати наступні дії:

1. Відкрити **CoDeSys**.
2. Вибрати меню **Файл – Створити**.
3. Настроїти цільову платформу:
   вибрати **None** або конфігурацію **3SCoDeSys SP PLCWint V2.4** (система виконання – емулятор промислового ПЛК з додатковими Web-можливостями). Натиснути **ОК**.
4. Залишити без зміни ім’я (PLC_PRG) і тип (програма) нового POU, вибрати

мову реалізації (наприклад, ST). Натиснути **OK**.

5. Набрати текст програми в текстовому (на мовах IL або ST) або графічному (на мовах SFC, LD, FBD або CFC) редакторі.
6. Відкомпіліювати проєкт: **Проєкт** – **Компіліювати** або **F11**.
7. Для тестування проєкту виконати його в режимі ***емуляції***.
8. Для візуального контролю функціонування ПЛК створити візуалізацію.
9. Перенести проєкт в ПЛК.

Примітка. Якщо проєкт створюється тільки для візуалізації або емуляції, то в якості конфігурації цільової платформи можна вибрати **None**.

Емуляція виконання проєкту

Для цього необхідно відкомпіліювати проєкт.

Вибрати меню **Онлайн** – відмітити **Режим емуляції**.

Вибрати меню **Онлайн** – **Підключення**.

Основні команди в режимі емуляції:

F5 **Старт** (емуляції),

Ctrl + F5 **Один цикл** (виконати),

Ctrl + F7 **Записати значення** (змінної),

Shift + F8 **Стоп** (емуляції),

Відключення.

Примітка. Для зміни значення змінної необхідно двічі клацнути по ній у вікні оголошення змінних. Якщо змінна має тип **BOOL**, то її значення зміниться на протилежне, для змінних інших типів у вікні, що з’явиться, слід набрати нове значення змінної. Натиснути **Ctrl + F7**.

Візуалізація виконання проєкту

Візуалізація призначена для графічного представлення об'єкту управління і безпосередньо пов'язана із створеною в CoDeSys програмою контролера. Редактор візуалізації CoDeSys надає набір готових графічних елементів, які можуть бути пов'язані відповідним чином зі змінними проекту.

Для створення візуалізації необхідно:

1) в Організаторі об’єктів відкрити закладку “**Візуалізація**”

2) **Проєкт → Об’єкт → Добавити**…

3) **Ім’я нової візуалізації**: набрати ім’я – **ОК.**

4) Нарисувати елементи візуалізації за допомогою наступних інструментів (рисунок 1).

Рисунок 1 – Інструменти візуалізації

5) Конфігурувати кожний елемент візуалізації:

а) подвійним клацанням по елементу візуалізації відкрити “**Конфігурування елементу (#...)**”;

б) вибрати **Кольори** (рисунок 2), натискаючи кнопки Заливка вибрати для елемента потрібні кольори;

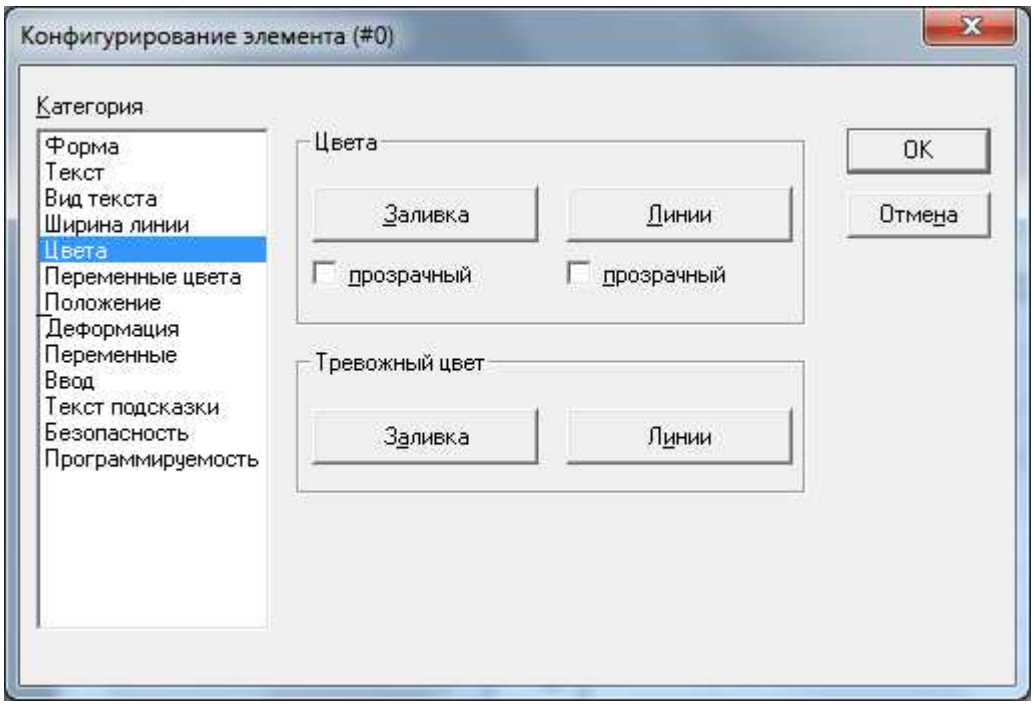

Рисунок 2 – Вибір кольору елемента

в) вибрати **Змінні – Зм. кольору**: **F2** – вибрати змінну, наприклад **x** (рисунок 3);

г) **Введення – Зм-а. перемикання: F2** – вибрати змінну (рисунок 4), **ОК.**

6) Підключити: **Онлайн – Підключення.**

7) Запуск проєкту: **Онлайн** – **Старт** або **F5**.

Примітка.

1. Перед підключенням переконайтеся, що обраний “**Режим емуляції**”.

2. З кожним елементом візуалізації можна зв’язати текст (категорія **Текст**) і спливаючу підказку (категорія **Текст підказки**).

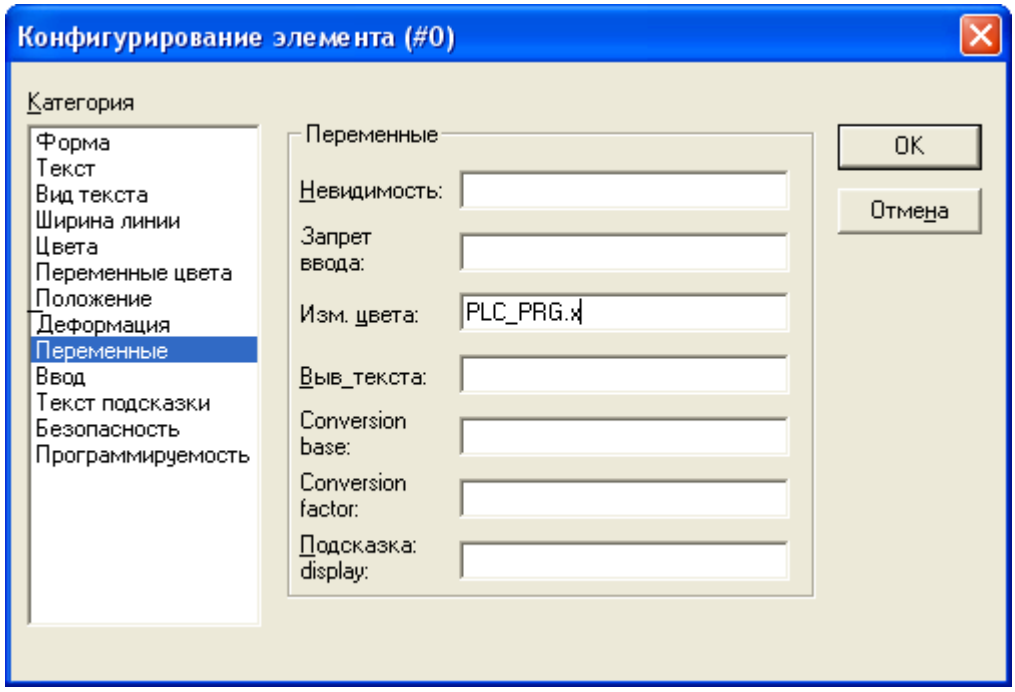

Рисунок 3 – Вікно конфігурування елемента візуалізації

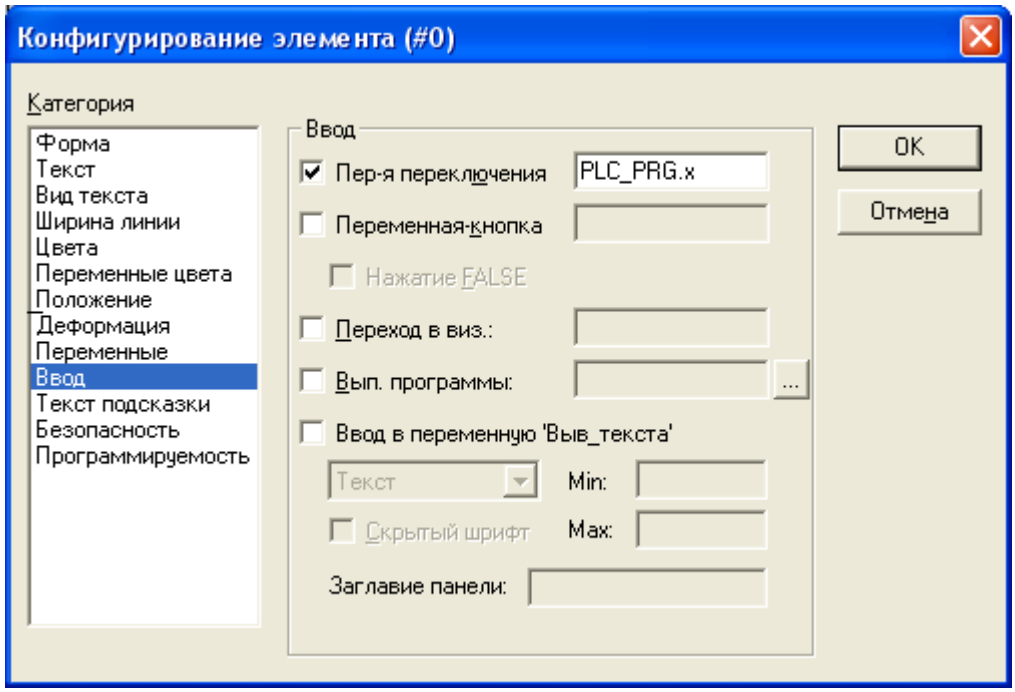

Рисунок 4 – Вибір змінної перемикання

Порядок виконання роботи

Постановка задачі. Створити ПЛК-програму для незалежного включення/виключення однієї лампочки двома перемикачами.

Рішення задачі. Створимо логічну функцію від двох аргументів, яка змінює своє значення на протилежне при кожній зміні тільки одного аргументу.

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Потрібна функція: $y = \overline{a} \wedge b \vee a \wedge \overline{b}$.

Запустимо **CoDeSys,** створимо проєкт командою **Файл→Створити** і вибираємо платформу **3SCoDeSys SP PLCWint V2.4**. Натискаємо **ОК** (рисунок 5).

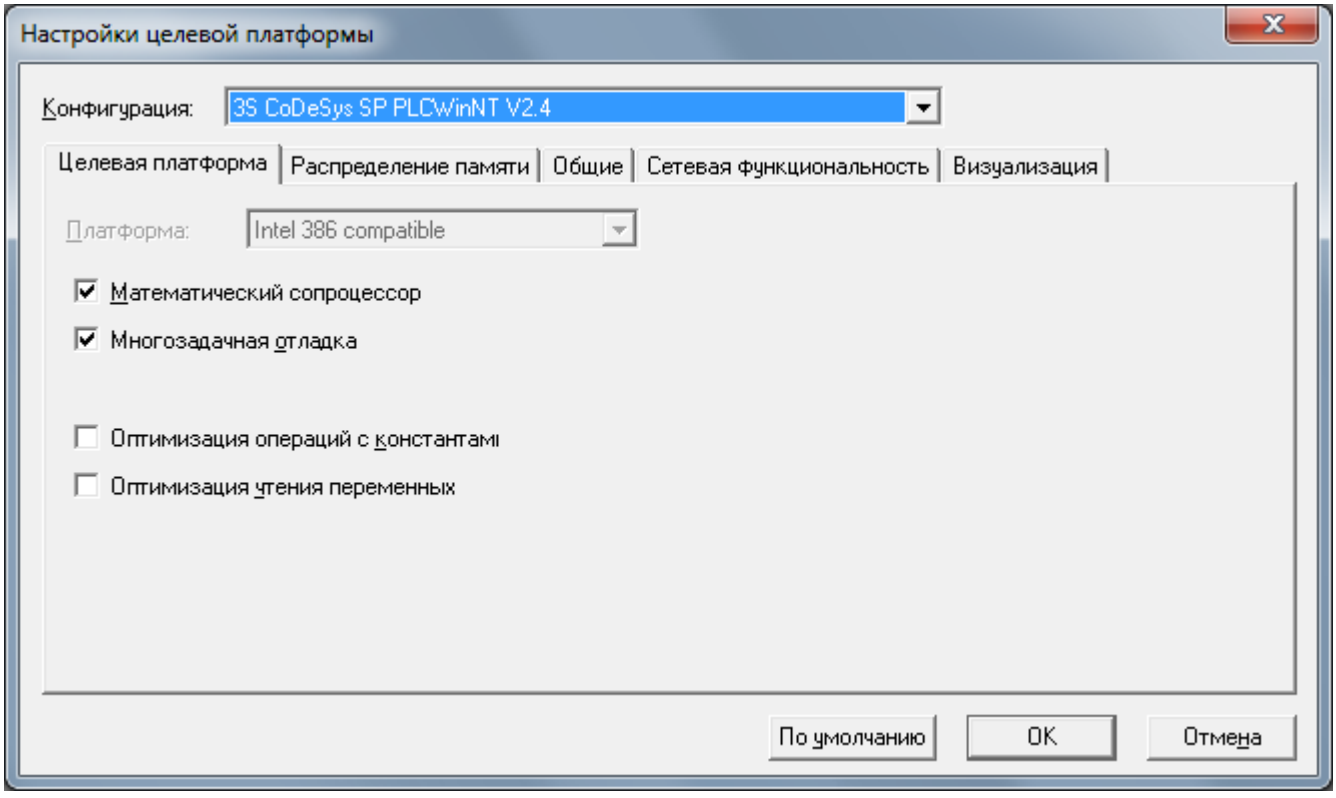

Рисунок 5 – Настройка цільової платформи

Створимо проєкт з одного компоненту (програми) з ім’ям **PLC_PRG** (за замовченням)**,** вибираємо мову **ST** (рисунок 6).

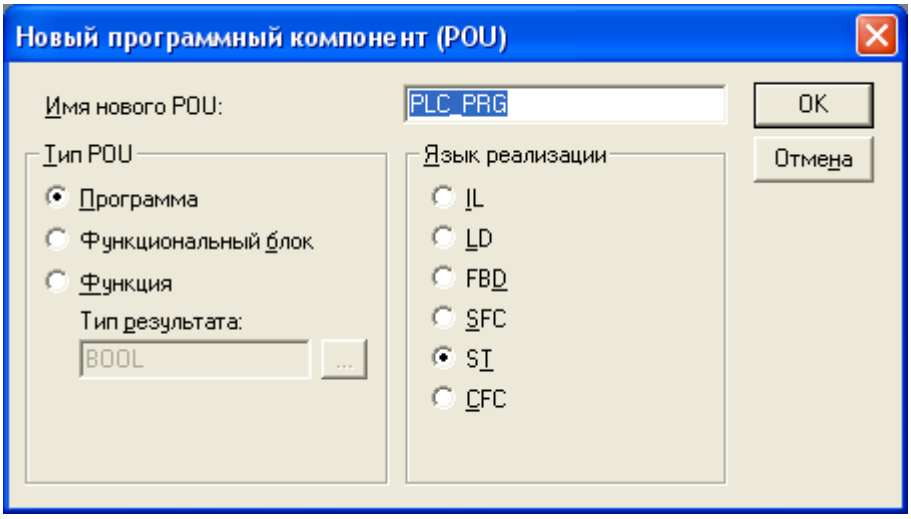

Рисунок 6 – Вікно вибору POU і мови програмування

Зберігаємо проєкт в папці командою **Файл→Зберігти як…** (рисунок 7).

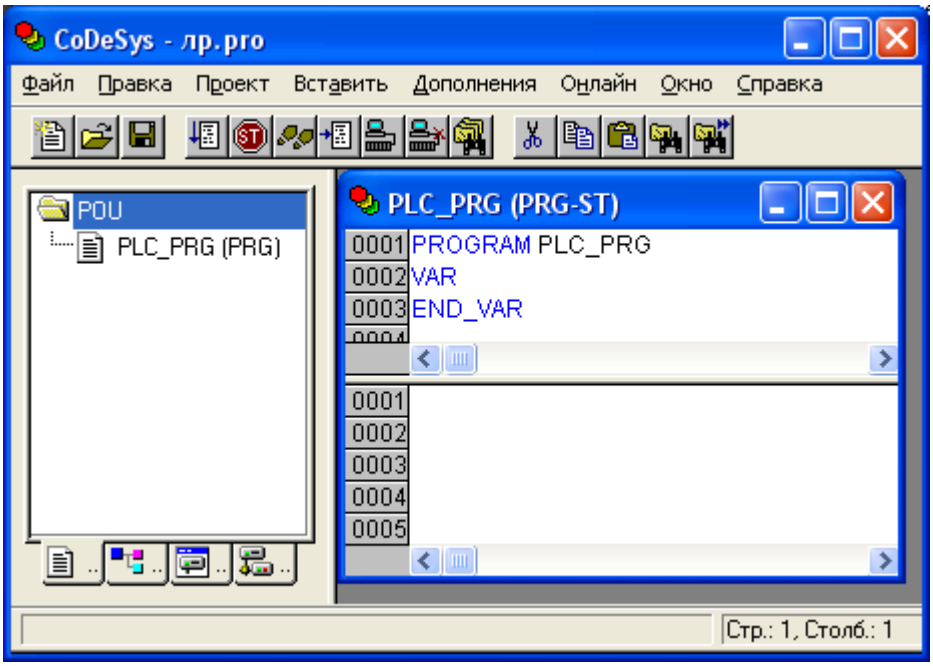

Рисунок 7 – Результат збереження проєкту

В розділі оголошення змінних програми **PLC_PRG** оголошуємо логічні змінні **a, b, y**. Для цього необхідно клацнути ПМК по полю цього розділу і вибрати з контекстного меню **Авто оголошення** (**Shift - F2**), у вікні, що з’явиться (рисунок 8), вказати ім’я змінної **a** і вибрати тип **BOOL,** натиснути **ОК** і змінна автоматично з’явиться в програмі. Аналогічно оголосити решту змінних.

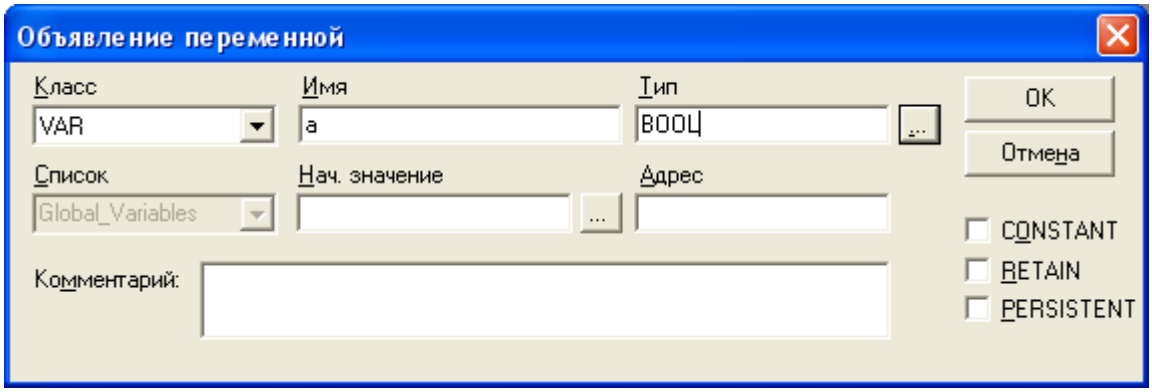

Рисунок 8 – Вікно оголошення змінної

Примітка. Оголосити змінну можна «вручну» в розділі оголошень. Якщо при наборі чергового оператора **CoDeSys** «зустріне» неоголошену змінну, то буде виведене вікно «**Объявление переменной**», в якому для змінної треба вибрати:

– «Класс» (**VAR, VAR_INPUT** тощо),

– «Тип» (**BOOL, BYTE** тощо)

– властивість (**CONSTANT, RETAIN, PERSISTENT**) (необов'язково).

В розділі операторів набираємо оператор розгалуження (**if**) (рисунок 9):

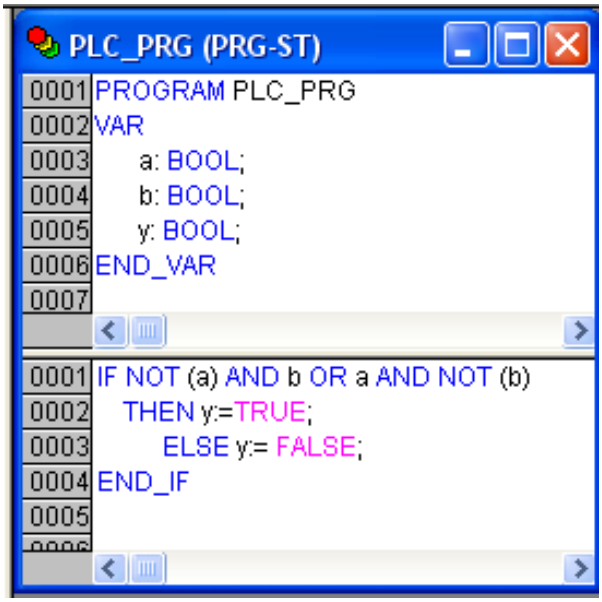

Рисунок 9 – Вікно програми

Компілюємо проєкт: **Проєкт – Компіліювати** або **F11**.

Результат компіляції буде виведено в нижньому вікні. В даному випадку побачимо (рисунок 10):

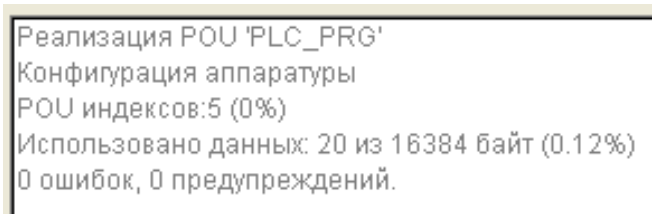

Рисунок 10 – Повідомлення компілятора

Примітка. Повідомлення про всі виявлені синтаксичні помилки виводяться червоним кольором; подвійне клацання по повідомленню позиціює курсор в позицію помилки.

Тестуємо проєкт в режимі емуляції. В меню **Онлайн** включити **Режим емуляції**. Потім **Підключення** і **Старт**.

В результаті з'явиться вікно емуляції (рисунок 11), в якому в розділах оголошення змінних і операторів указані початкові значення змінних.

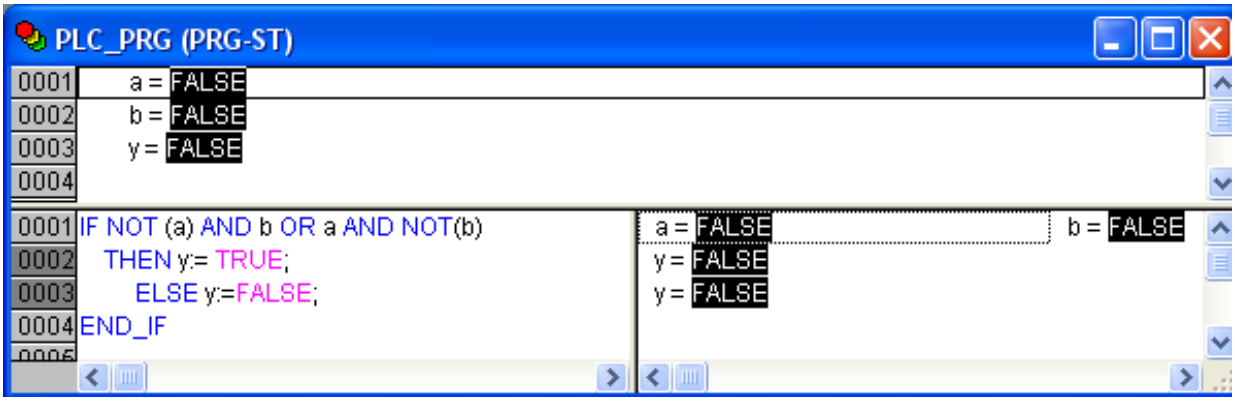

Рисунок 11 – Вікно емуляції

Відключити режим емуляції: **Онлайн** – **Відключення**.

Створюємо візуалізацію проєкту (рисунок 12):

Рисунок 12 – Візуалізація проєкту

З першим прямокутником зв’язуємо змінну **a**, з другим – **b**, а з колом – **y**.

Для «оживлення» візуалізації добавимо в проєкт змінну **x**, яка в кожному робочому циклі буде збільшуватися на 1 по mod 4, змінній **a** присвоємо нульовий біт **x**, а **b** – перший (рисунок 13).

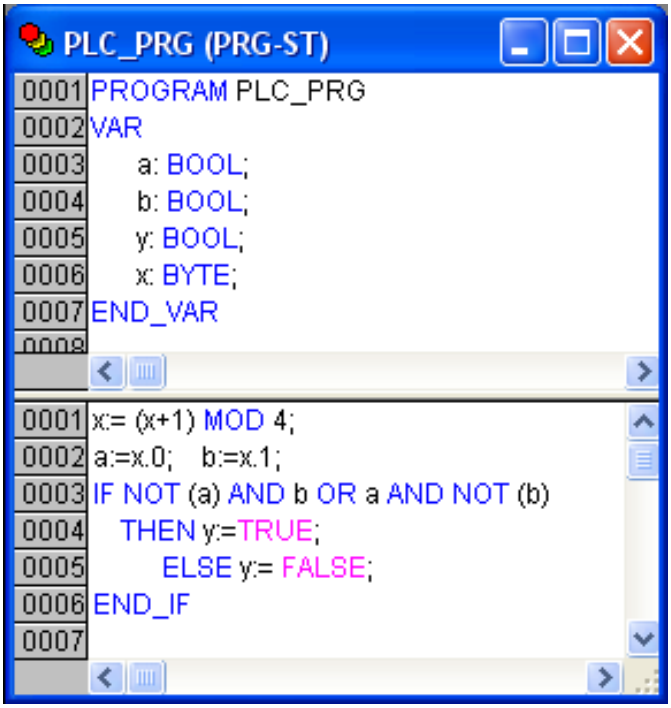

Рисунок 13 – Змінення програми

Таким чином, змінні **a** і **b** приймуть по черзі всі можливі значення. Кадр візуалізації проєкту показаний на рисунку 14.

Рисунок 14 – Кадр візуалізації проєкту

На цьому кадрі обидва перемикачі замкнуті, а лампочка не горить.

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

Варіант	Функція	Варіант	Функція
1	$y = \overline{a} \wedge b \vee \overline{a} \wedge b$	11	$y = \overline{a} \wedge \overline{b} \vee \overline{a} \wedge \overline{b}$
2	$y = \overline{a} \wedge \overline{b} \vee a \wedge b$	12	$y = \overline{a} \wedge \overline{b} \vee \overline{a} \wedge b$
3	$y = a \wedge \overline{b} \vee \overline{a} \wedge b$	13	$y = \overline{a} \wedge \overline{b} \vee a \wedge \overline{b}$
4	$y = a \wedge \overline{b} \vee a \wedge \overline{b}$	14	$y = a \wedge \overline{b} \vee \overline{a} \wedge \overline{b}$
5	$y = a \wedge b \vee \overline{a} \wedge \overline{b}$	15	$y = \overline{a} \wedge b \vee \overline{a} \wedge \overline{b}$
6	$y = \overline{a} \wedge b \vee a \wedge b$	16	$y = \overline{a} \wedge \overline{b} \vee a \wedge b$
7	$y = a \wedge \overline{b} \vee a \wedge b$	17	$y = \overline{a} \wedge b \vee \overline{a} \wedge b$
8	$y = a \wedge b \vee \overline{a} \wedge b$	18	$y = \overline{a} \wedge \overline{b} \vee a \wedge b$
9	$y = a \wedge b \vee a \wedge \overline{b}$	19	$y = a \wedge \overline{b} \vee \overline{a} \wedge b$
10	$y = a \wedge b \vee a \wedge b$	20	$y = a \wedge \overline{b} \vee a \wedge \overline{b}$

Контрольні питання

1. З яких етапів складається розробка проєкту в CoDeSys?

7. Для чого призначена візуалізація?

8. Як конфігурувати елемент візуалізації?

9. Як створити підказку для елемента візуалізації?

11. Як скомпілювати проект?

12. Для чого призначена емуляція?

13. Як мені увімкнути або вимкнути режим емуляції?

14. Як оголосити змінні в програмі?

15. Які значення може приймати логічна змінна?

16. Які ви знаєте логічні оператори, який у них пріоритет?

ЛАБОРАТОРНА РОБОТА № 3

Тема: РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ТРАНСПОРТЕРАМИ

Мета роботи: отримати практичні навички проєктування ПЛК-програми керування електроприводами транспортерів засобами мови LD стандарту МЕК 61131-3; ознайомитися із засобами бібліотек в CoDeSys.

Постановка задачі: розробити систему керування електроприводами транспортерів.

Короткі теоретичні відомості

Мова LD

Мова релейних або релейно-контактних схем (РКС) - графічна мова, який реалізує структури електричних ланцюгів.

Найкраще LD підходить для побудови логічних перемикачів, але досить легко можна створювати і складні ланцюга - як в FBD. Крім того, LD досить зручний для управління іншими компонентами POU.

LD працює з послідовністю ланцюгів, кожен з яких містить логічний або арифметичний вираз, виклик функціонального блоку, перехід або інструкцію повернення.

Діаграма LD

Діаграма LD складається з ряду ланцюгів.

Ліворуч і праворуч схема обмежена вертикальними лініями - шинами живлення. Між ними розташовані ланцюги, які утворені контактами і обмотками реле, за аналогією зі звичайними електронними ланцюгами.

Зліва будь-який ланцюг починається набором контактів, які посилають зліва направо стан "ON" або "OFF", відповідні логічним значенням ІСТИНА або ХИБНІСТЬ. Кожному контакту відповідає логічна змінна. Якщо змінна має значення ІСТИНА, то стан передається через контакт. Інакше праве з'єднання отримує значення вимкнено ("OFF").

Приклад типової LD-схеми в CoDeSys показаний на рисунку 1.

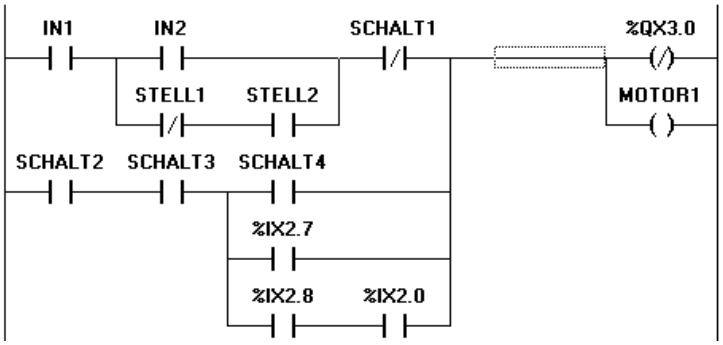

Рисунок 1 – LD-схема

Контакт

Контакти позначаються двома паралельними лініями і можуть мати стани "ON" або "OFF". Ці стани відповідають значенням ІСТИНА або ХИБНІСТЬ. Кожному контакту відповідає логічна змінна. Якщо значення змінної ІСТИНА, то контакт замкнутий.

Контакти можуть бути з'єднані паралельно, тоді з'єднання передає стан "ON", коли хоча б одна з гілок передає "ON".

Якщо контакти з'єднані послідовно, то для того, щоб з'єднання передало "ON", необхідно, щоб обидва контакти передавали "ON". Це відповідає електричній паралельній і послідовній схемі.

Контакт може бути інверсним. Такий контакт позначається за допомогою символу | / | і передає стан "ON", якщо значення змінної ХИБНІСТЬ.

Обмотка

У правій частині схеми може бути будь-яка кількість обмоток (реле), які позначаються круглими дужками (). Вони можуть з'єднуватися тільки паралельно. Обмотка передає значення з'єднання зліва направо і копіює його в відповідну логічну змінну.

В цілому ланцюг може бути або замкнутий (ON), або розімкнутий (OFF). Це як раз і відображається на обмотці і відповідно на логічній змінній обмотки (ІСТИНА / ХИБНІСТЬ).

Обмотки також можуть бути інверсними (в прикладі% QX3.0). Якщо обмотка інверсна (позначається символом (/)), тоді в відповідну логічну змінну копіюється інверсне значення.

Функціональні блоки в LD

Крім контактів і обмоток, в LD можна використовувати функціональні блоки і програми. Вони повинні мати логічні вхід і вихід і можуть використовуватися так само, як контакти.

SET і RESET обмотка

Обмотки можуть бути з "самофіксацією" типів SET і RESET. Обмотки типу SET позначаються літерою "S" всередині круглих дужок (S). Якщо відповідна цій обмотці змінна приймає значення ІСТИНА, то вона назавжди (до скидання R) зберігає його.

Обмотки типу RESET позначаються літерою R. Якщо відповідна змінна приймає значення ХИБНІСТЬ, то вона назавжди (до установки S) зберігає його.

Порядок виконання роботи

Постановка задачі. Необхідно створити систему керування електроприводами горизонтального **А** і похилого **В** транспортерів для транспортування подрібненої сировини (гумової крихти) в приміщенні по переробці автопокришок (рисунок 2).

Спочатку необхідно скласти словесний опис проектованої системи. На цій стадії слід виявити кількість і технічні характеристики вхідних елементів і виконавчих механізмів, спираючись на відомі рішення подібних завдань і особистий досвід, побажання технологів і обслуговуючого персоналу, вимоги безпеки.

Припустимо, в результаті цих дій виявили, що необхідно мати на пульті управління два приймальні елементи: кнопки «Пуск» і «Стоп», працюючі з самоповерненням, т. е. без фіксації включеного стану і для приводу транспортерів

два виконавчі механізми: **М1** і **М2.** Проектовану систему можна віднести до систем м'якого реального часу. В якості контролера можна використати ОВЕН ПЛК 100PL.

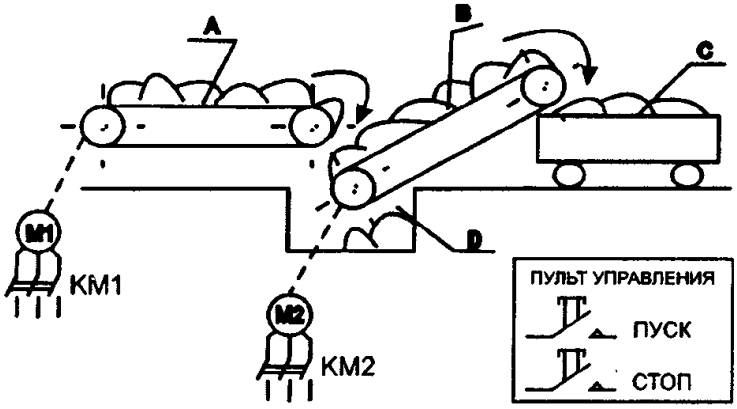

Рисунок 2 – Двохтранспортерна лінія:

А - горизонтальний транспортер; **В** - нахилений транспортер;

С - транспортний візок; **D** - сировинний бункер;

КМ1 і **КМ2** - контакти магнітних пускачів

При пуску лінії першим повинен включитися **М2** (щоб не було завалу в сировинному бункері) і через 10 сек (за порадою технологів) двигун **М1**.

При нормальному режимі роботи нахилений транспортер повинен працювати 10 хв., але першим за 20 сек до зупинки **М2** повинен вимкнутися **М1**, щоб за цей час розвантажити сировинний бункер і полегшити подальший запуск лінії.

У випадки малої кількості сировини оператор може достроково, натиснути кнопку «Стоп», зупинити процес. При цьому першим повинен зупинитися **М1** і через 20 с - **М2.** При необхідності (аварійна ситуація!) відразу зупинити **М1** і **М2**, короткочасно натиснувши обидві кнопки «Пуск» і «Стоп» одночасно. Повторне (можливо хибне) натиснення цих кнопок не повинно призводити до включення транспортерів. Повторний запуск транспортерів допускається не менше, чим через 10 хв. з метою охолодження двигунів **М1** і **М2** або для з'ясування причин аварійної зупинки і їх усунення.

Схема роботи представлена у вигляді тимчасової діаграми на рисунку 3.

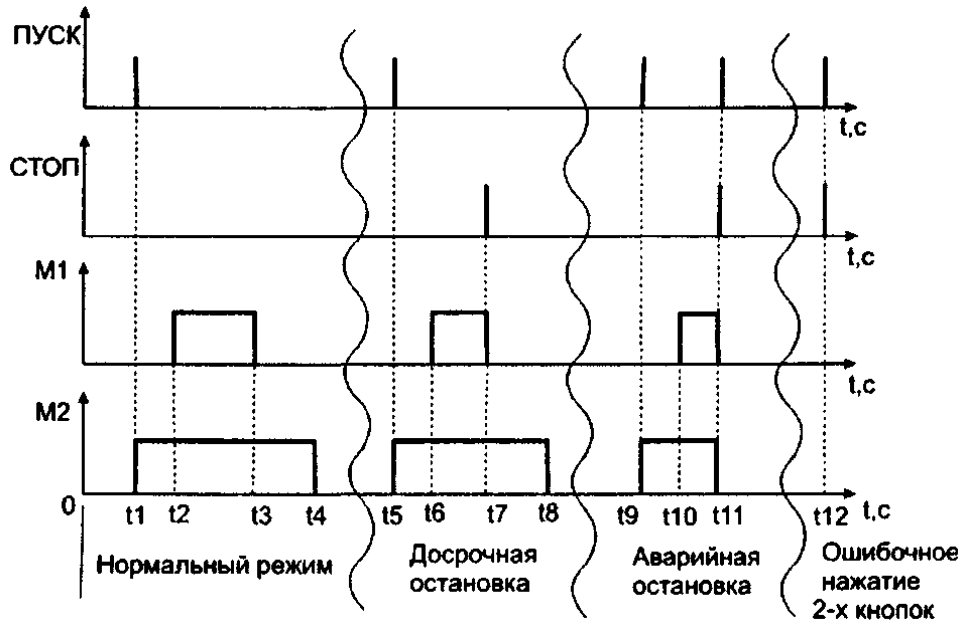

Рисунок 3 – Тимчасова діаграма станів вхідних і вихідних елементів проектованої системи

Очевидно:

t2 - t1 = t6 - t5 = t10 - t9 = 10 с; (F4)

t4 - t3 = t8 - t7 = 20 с; (F7)

t4 - t1 = 10 хв. = 600 с; (F3)

t3 - t2 = 600 - 10 - 20 = 570 с. (F5=F3 - F4 - F7)

Опишемо режими роботи по цій діаграмі і будемо одночасно проєктувати схему на мові LD.

Рішення задачі

Налаштування цільової платформи

При створенні проєкту вибираємо цільову платформу **3SCoDeSys SP PLCWint V2.4.**

Головна програма PLC_PRG POU

Наступне діалогове вікно визначає тип першого програмного компонента (**POU**). Виберете мову реалізації **LD** і збережете запропоновані за умовчанням тип компонента - **Програма** і ім'я - **PLC_PRG** (рисунок 4).

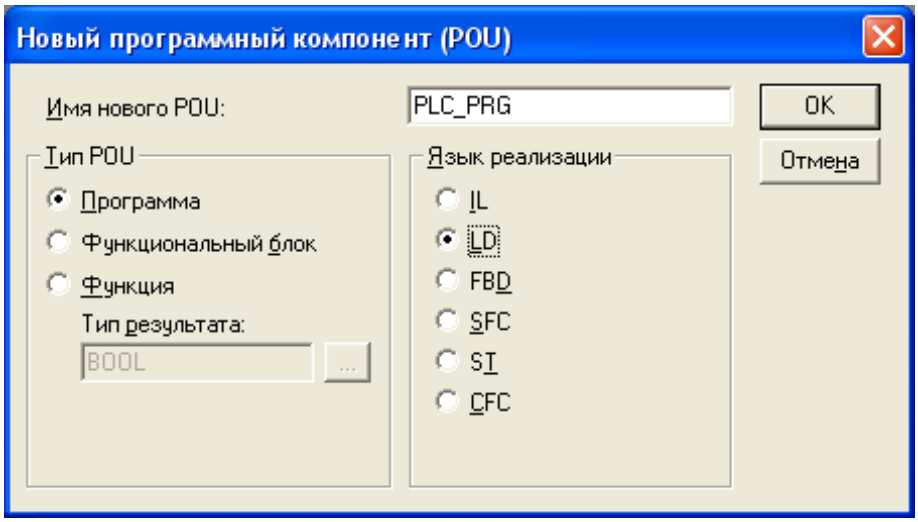

Рисунок 4 – Вибір POU і мови програмування

Підключення standard.lib

Для створення таймерів і тригерів необхідно підключити стандартну бібліотеку. Для цього відкрийте менеджер бібліотек командами **Вікно - Менеджер бібліотек**. Виберіть **Вставка - Додати бібліотеку**. Повинне відкритися діалогове вікно вибору файлів. Виберете **STANDARD.lib** зі списку бібліотек (рисунок 5).

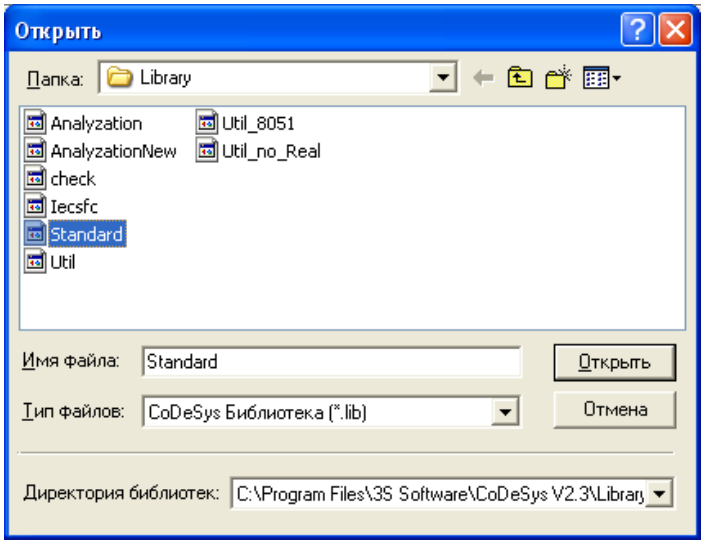

Рисунок 5 – Підключення STANDARD.lib

Перший ланцюг. Вставляємо в ланцюг елементи: Контакт (ПКМ), **ТР-таймер (**ПКМ, Функціональний блок, Timer, TP(FB), OK), Обмотка (ПКМ).

Міняємо ??? кожного елемента на PUSK**,** **F1** і K1 відповідно. Н вхід PT **TP-таймера** подаємо 100 мс (рисунок 6).

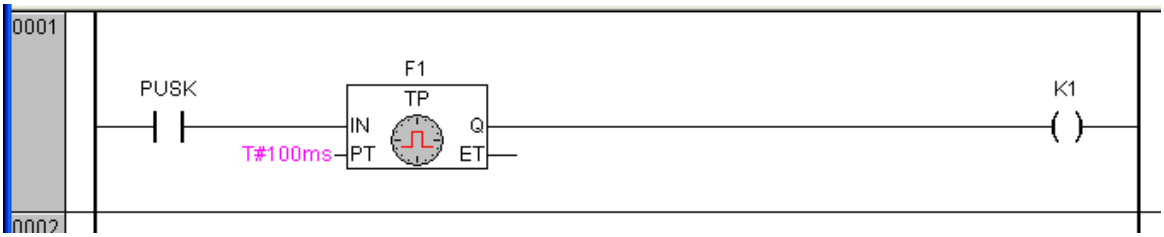

Рисунок 6 **–** Перший ланцюг LD- схеми

У момент **t1** оператор натиснув кнопку **PUSK**, і через **TP-таймер F1** короткочасно спрацювало віртуальне реле **K1** (рисунок 6). До речі, цей таймер потім при підключенні кнопки до реального ПЛК захистить його від так званого «брязкоту контактів», який має місце в механічних контактних елементах. Під час цього брязкоту виникає хаотична група імпульсів, що у ряді випадків може привести до неправдивих спрацьовувань ПЛК.

TP-таймер же реагує на перший імпульс, ігноруючи наступні, якщо ті укладаються в часовий інтервал уставки по його входу PT (рисунок 6). Зазвичай досить прийняти уставку близько десятків мілісекунд, так як за технічними умовами на контактні вироби тривалість брязкоту не повинна перевищувати 1мс. Але значення PT, в загальному випадку, не повинне перевищувати час прогону програми.

Другий ланцюг. Складові елементи: Контакт, **RS**-тригер (ПКМ, Функціональний блок…, Bistable Function Blocks, SR(FB), OK), TP-таймер, Обмотка. Відповідні змінні: K1**,** F2**,** F3**,** M2 (рисунок 7).

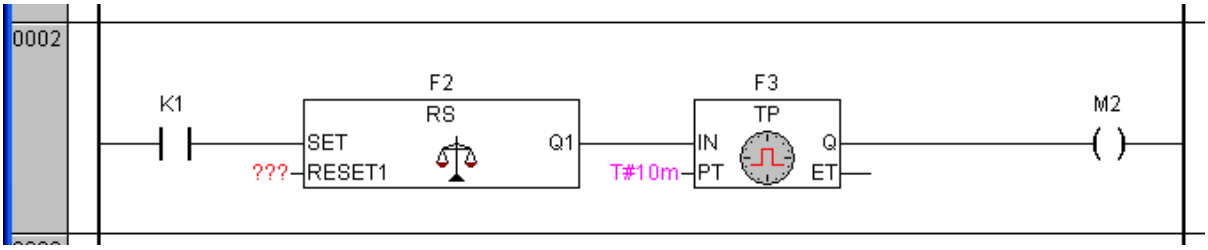

Рисунок 7 **–** Другий ланцюг LD- схеми

Перший ланцюг імітує роботу кнопки з самоповерненням, тобто при натисненні на кнопку **PUSK** короткочасно (на період дії імпульсу з **ТР-таймера F1**) спрацюють контакти цього реле. Тепер необхідно зафіксувати це короткочасне спрацьовування. З цією метою в другому ланцюзі поставимо RS-тригер F2**.** При замиканні контакту К1 на вході SET цього FB на його виході Q1 з'явиться сигнал, який запускає ТР-таймер F3 з уставкою РТ на 10 мін (600 с). Можна встановити за бажанням будь-який варіант РТ. Це забезпече включення в той же момент tl двигуна нахиленого транспортера.

Третій ланцюг. Складові елементи: Контакт, **TON-** таймер, **ТР-таймер**, Обмотка. Відповідні змінні: **M2**, **F4**, **F5**, **M1** (рисунок 8).

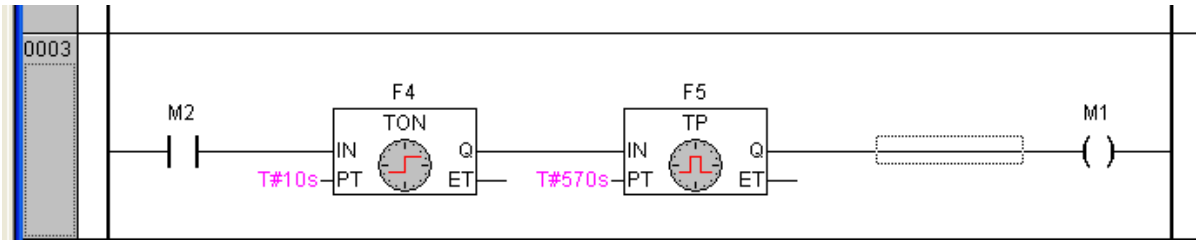

Рисунок 8 – Третій ланцюг LD- схеми

У третьому ланцюзі віртуальний контакт реле **M2** через **TON-** таймер **F4** із затримкою на 10 с запустить **ТР-таймер F5** з уставкою **РТ** на 9,5 хв. Цей ланцюг забезпечить у момент **t2** запуск **M1** і його відключення у момент **t3**, тобто за 20 с до зупинки **M2.** Початковий фрагмент проектованої системи зображений на рисунку 9.

Поки ??? на вході **RESERT1** в **RS**-тригері залишимо без відповіді. Формально нормальний режим роботи транспортерів в інтервалі **0**...**t4** виконаний.

Дострокова зупинка

Четвертий ланцюг. Складові елементи: Контакт, **ТР-таймер**, Обмотка. Відповідні змінні: **STOP**, **F6**, **K2** (рисунок 10).

Необхідно передбачити можливість дострокової зупинки процесу транспортування сировини. З цією метою в четвертому ланцюзі встановимо кнопку **STOP** і по аналогії з вищеописаною схемою **ТР-таймер F6**.

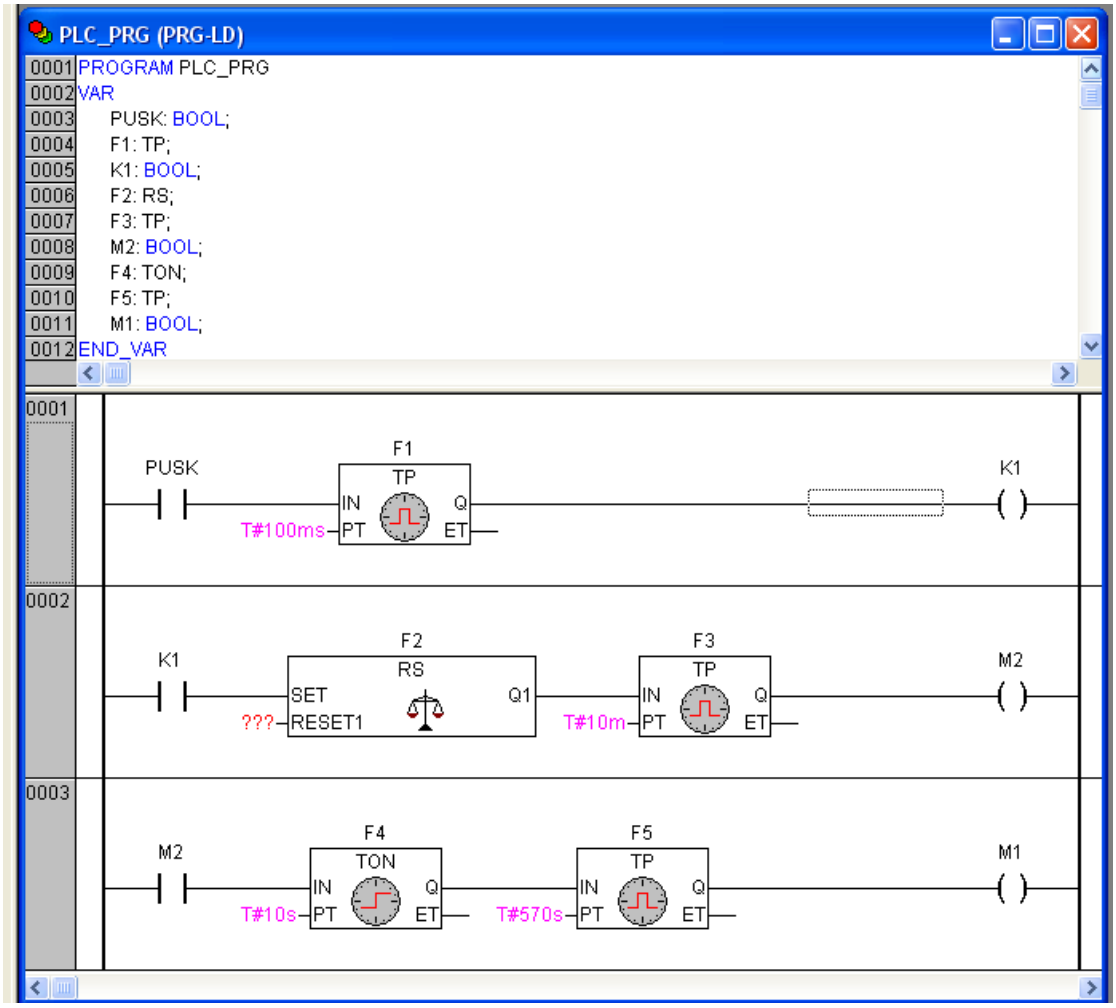

Рисунок 9 – Початковий фрагмент проектованої системи

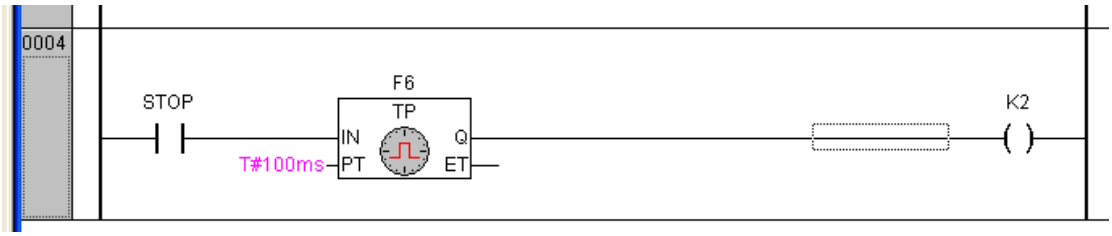

Рисунок 10 – Четвертий ланцюг LD- схеми

Необхідно передбачити можливість дострокової зупинки процесу транспортування сировини. З цією метою в четвертому ланцюзі встановимо кнопку **STOP** і по аналогії з вищеописаною схемою **ТР-таймер F6**.

П'ятий ланцюг. Складові елементи: Контакт (**K2**), Паралельний контакт (**Y1**), Інверсний контакт (???), Обмотка(**Y1**) (рисунок 11).

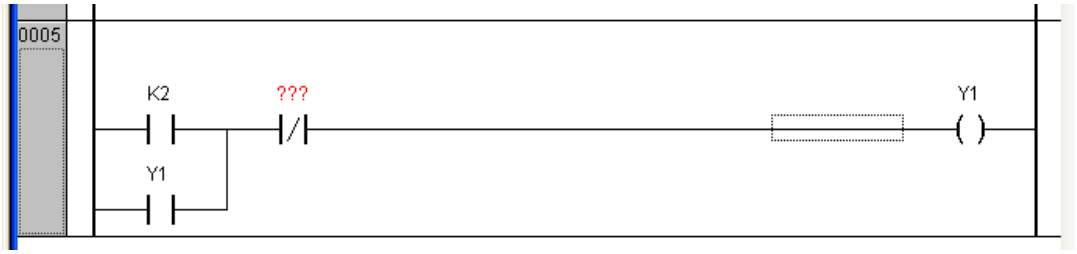

Рисунок 11 – П'ятий ланцюг LD- схеми

У момент **t7** при натисненні на кнопку **STOP** короткочасно спрацьовує реле **K2**, контакт якого в наступному п'ятому ланцюзі викликає спрацьовування реле **Y1**.

Це реле стає на самоблокування і в наступному циклі прогону програми, тобто з позиції системи м'якого реального часу практично в той же момент **t7** повинне зупинити **M1**. З цією метою в третій ланцюг поставимо розімкнений контакт **Y1**. Крім того, потрібно через 20 с відключити **M2**.

Шостий ланцюг. Елементи: Контакт(**Y1**), **TON**-таймер (**F7**), Обмотка (**Y2**).

Знадобиться новий шостий ланцюг (рисунок 12), в якому **Y1** через **TON**-таймер **F7** запускає віртуальне реле **Y2**, яке своїм розмикаючим контактом в другому ланцюзі виконає цю процедуру і в момент t8**,** тобто через 20 с після зупинки M1 відкючиться M2. Природно, колись доведеться зняти самофіксацію з реле Y1. Для цього поставимо розмикаючий контакт в ланцюг з котушкою Y1. Але ім'я цього елемента залишимо доки під ???.

Виконали і другий режим дострокової зупинки.

Ще необхідно вирішити питання з перезапуском **RS-тригера** і аварійною зупинкою **M1** і **M2**.

Створимо ще один **сьомий ланцюг.** Елементи: Контакт (**K1**), Контакт (**K2**), **TP-таймер** (**F8**), Обмотка (рисунок 13).

Контактами **K1** і **K2**, які спрацьовують у момент **t11** при натисненні на кнопки **PUSK** і **STOP** через **TP-таймер F8** запустимо реле **h**, а ім'я цього реле поставимо на вході **RESET1 RS**-тригера **F2**.

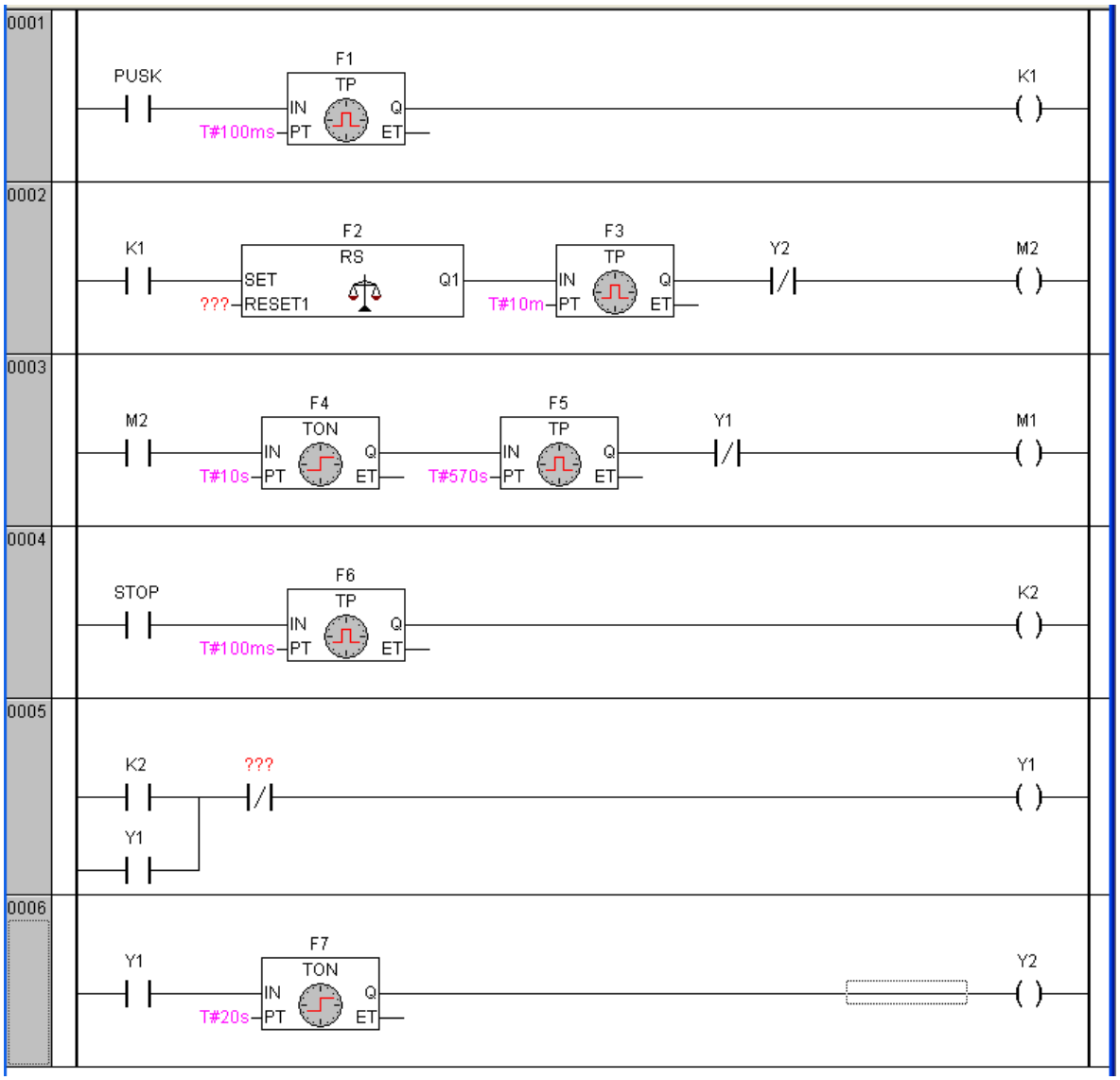

Рисунок 12 – Розвиток початкового фрагмента

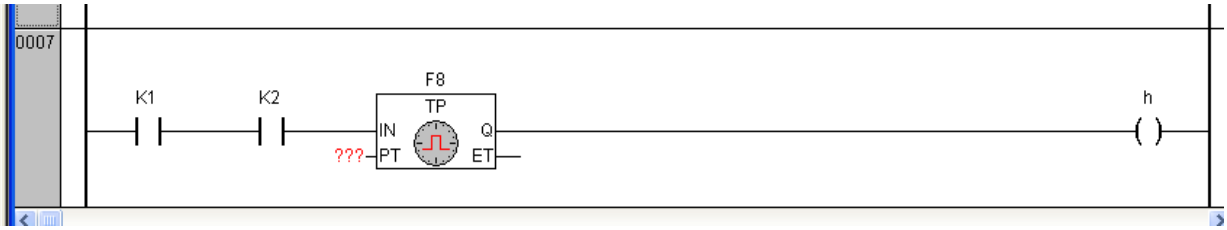

Рисунок 13 – Сьомий ланцюг LD-схеми

Крім того, розмикаючий контакт цього реле в той же момент **t11**, а точніше в наступному циклі прогону програми зніме блокування з реле **Y1** і знеструмить двигуни **M1** і **M2.** До речі, виконати точне одномоментне натиснення і включення механічних кнопок **PUSK** і **STOP** під силу не кожному операторові. Тут допоможуть таймери **F1** і **F6**, які не лише захищають ПЛК від брязкоту контактів,

але і усувають наслідки можливої асинхронності в діях оператора при натисненні на кнопки. Уставка **ТР-таймера F8** має бути не менше уставки запущеного раніше **ТР-таймера F3.** За цей час знову запустити технологічний процес не можна. (Потрібно встановити і усунути причину аварійної зупинки!)

Отримали завершальний варіант проектованої системи (рисунок 14).

Якщо запустити цю програму в режим емуляції, то при першому запуску все спрацює, як і планувалося.

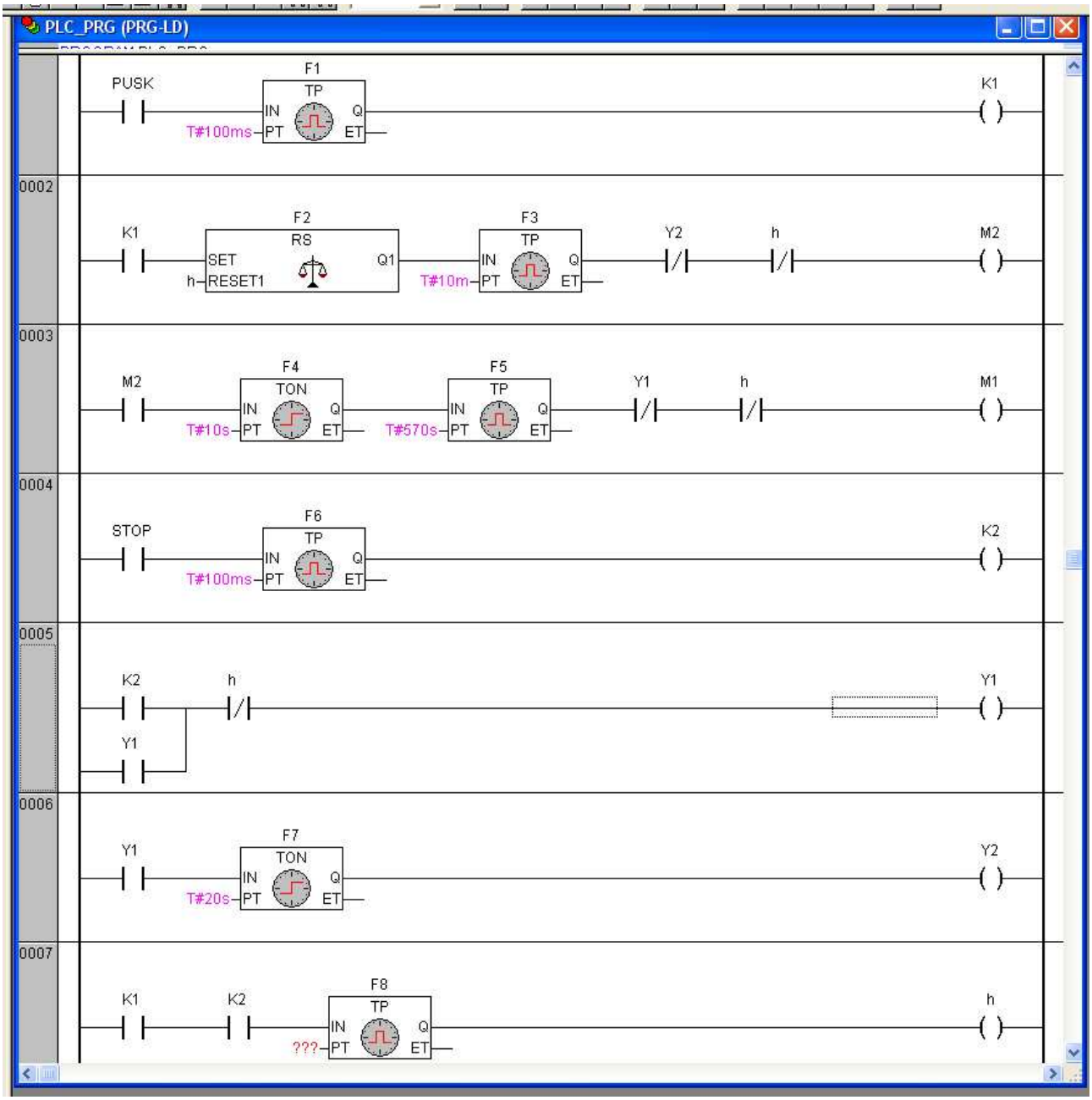

Рисунок 14 – Завершальний варіант системи

Але спроба повторного запуску в першому і другому режимах (рисунок 14) не дасть бажаного результату. Оскільки, вихід **Q1 RS**-тригера після вступу сигналу по **SET** входу залишається в стані TRUE, поки не отримає сигнал зброcу на вхід **RESET1**. А цей сигнал поступить тільки в режимі аварійної зупинки, тобто коли будуть натиснуті кнопки **PUSK** і **STOP**. Отже, потрібно добитися, щоб при кожному натисненні на кнопку **PUSK** проводилося короткочасне подання сигналу на вхід **RESET1**.

З цією метою створимо ще один (восьмий) ланцюг.

Восьмий ланцюг. Елементи: Контакт (**K1**), **ТР-таймер** (**F9**), Обмотка (**K3**).

У цьому ланцюзі контакт **K1**, що спрацьовує при натисненні кнопки **PUSK**, запустить **ТР-таймер F9** з витримкою часу значно менше, ніж у таймера **F1** (рисунок 15). Спрацює реле **K3** і своїм замикаючим контактом в наступному циклі приведе до короткочасного спрацьовування котушки реле **h**, встановленої в сьомому ланцюзі. Це реле вже в *наступному* циклі прогону програми подасть своїм замикаючим контактом (на схемі його немає!) короткочасний сигнал TRUE по входу **RESET1 RS**-тригера, підготувавши тим самим його до прийому сигналу по входу **SET**. Тому уставка **PT** у **F1** повинна бути, принаймні, більше аналогічної уставки для **F9** на подвійну тривалість циклу сканування. Інакше **RS**-тригер не спрацює.

Питання безпеки

З метою безпечної роботи технологічної лінії бажано кнопку **STOP** вибирати з розмикаючим контактом, тобто у випадку обриву лінії зв'язку цієї кнопки з ПЛК з'явиться сигнал, потребуючий втручання технологів. З позицій безпечного обслуговування подібного технологічного процесу можна стверджувати, що краще не запустити, чим вчасно не зупинити!

Тоді четвертий ланцюг в схемі за рисунком 15 можна замінити, як показано на рисунку 16. Знадобиться ще один функціональний блок **F-** тригер (F10).

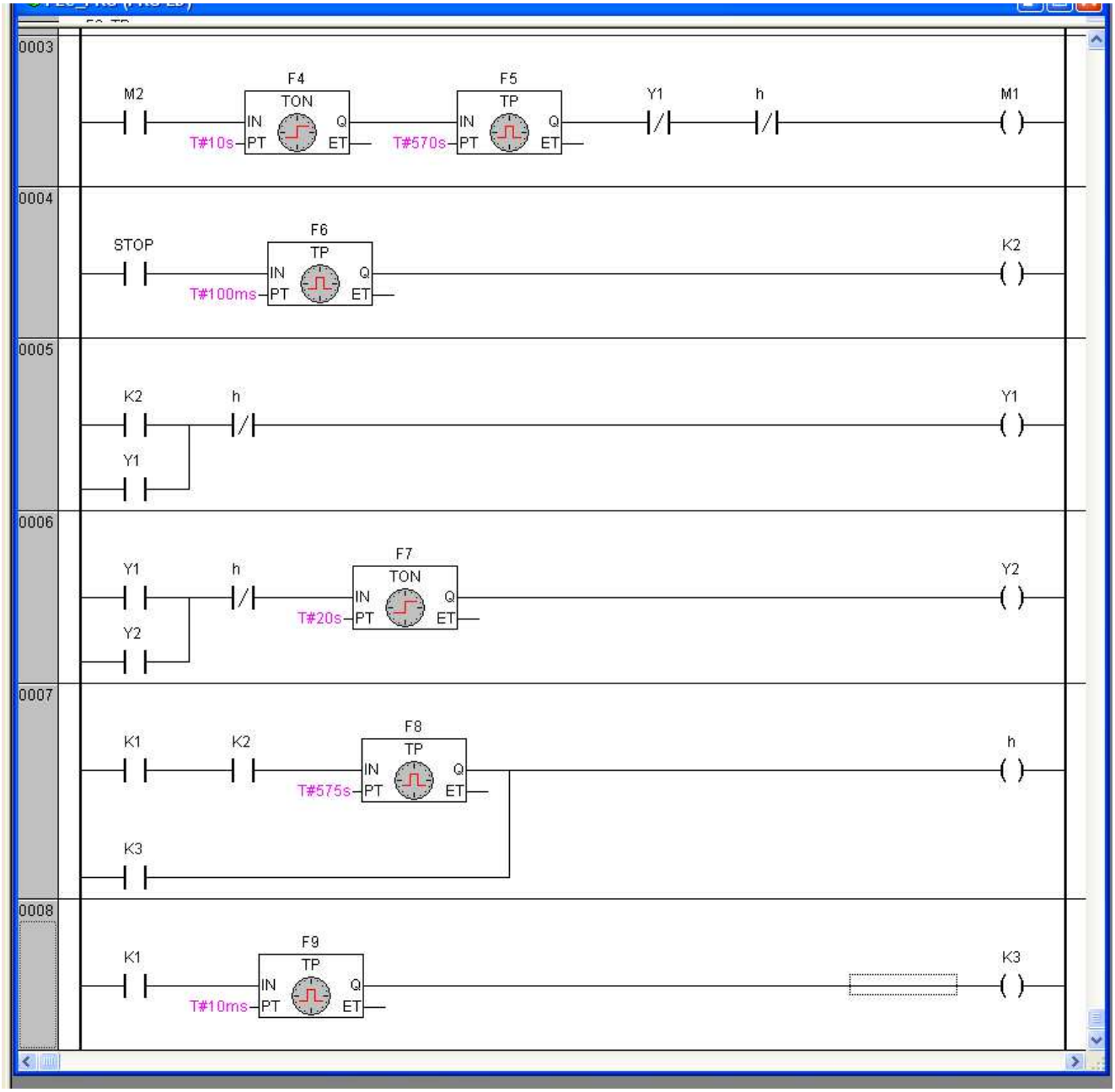

Рисунок 15 – Остаточний варіант проектованої системи

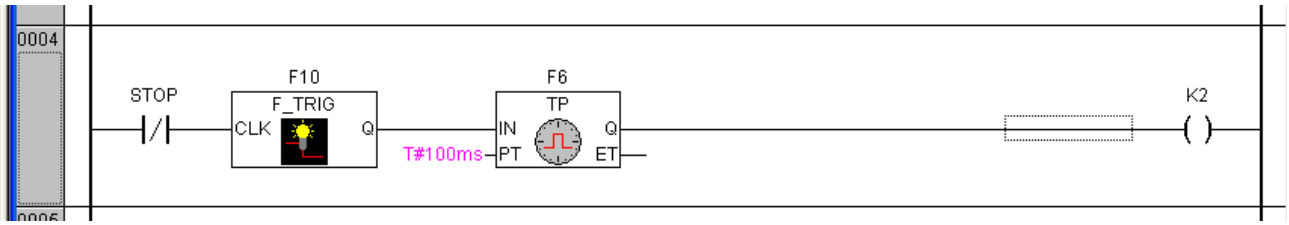

Рисунок 16 – Фрагмент схеми з розмикаючою кнопкою STOP

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№ варіанту	Уставки РТ таймерів					
	F1, мс	F3, мін	F4, с	F6, мс	F7, с	F9, мс
1	100	4	7	100	14	10
2	100	5	8	100	16	10
3	100	6	9	100	19	10
4	100	7	11	100	22	10
5	100	8	12	100	24	10
6	100	9	13	100	26	10
7	100	10	14	100	28	10
8	100	1	5	100	10	10
9	100	2	6	100	12	10
10	100	3	7	100	13	10
11	100	4	6	100	12	10
12	100	5	7	100	14	10
13	100	6	8	100	16	10
14	100	7	10	100	20	10
15	100	8	11	100	22	10
16	100	9	12	100	24	10
17	100	10	13	100	26	10
18	100	11	14	100	28	10
19	100	12	15	100	30	10
20	100	13	16	100	32	10

Контрольні питання

1. Мова LD і LD-діаграма.
2. Контакти і обмотки реле. Способи з'єднання.
3. Функціональні блоки в LD.
4. SET і RESET обмотка.

ЛАБОРАТОРНА РОБОТА № 4

Тема: РОЗРОБКА ВІЗУАЛІЗАЦІЇ СИСТЕМИ КЕРУВАННЯ ТРАНСПОРТЕРАМИ

Мета роботи: отримати практичні навички в створенні візуалізації системі керування транспортерами.

Постановка задачі: Розробити інтерфейс оператора (візуалізацію) розробленої системи керування транспортерами.

Порядок виконання роботи

Відкрити проект, розроблений на попередній лабораторній роботі.

Створення візуалізації

Для того, щоб створити візуалізацію, виберіть вкладку **Візуалізація** в Організаторові об'єктів. Натисніть ПМК і виберіть команду **Додати об'єкт**. У вікні, що з'явилося, введіть будь-яке ім'я для візуалізації, наприклад **transporter**, натискаємо ОК (рисунок 1).

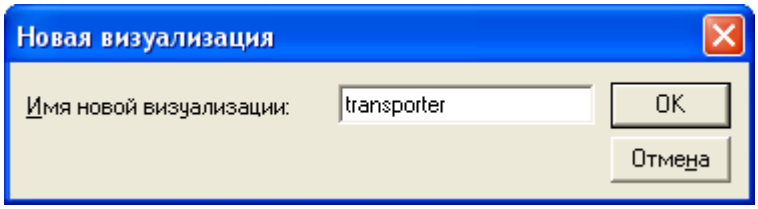

Рисунок 1 – Діалог для створення нової візуалізації

Відкриється вікно, в якому і створюватиметься графічний об'єкт управління.

Бажано в процесі створення візуалізації орієнтуватися на схему роботи транспортера, описану в лабораторній роботі 3, щоб графічний об'єкт вийшов реалістичнішим і зрозумілішим.

Вставка графічних елементів у візуалізацію

Розпочнемо створення візуалізації із зображення площадки, на якій згодом і знаходитимуться усі інші механізми.

Для цього виберемо з Панелі інструментів елемент «Прямокутник». Клацаємо 1ЛКМ на робочому полі і, утримуючи ЛКМ, розтягуємо прямокутник

до необхідних розмірів і переміщуємо його в лівий нижній кут вікна візуалізації. Натискаємо 2ЛКМ на прямокутнику, що вийшов, і вибираємо категорію **Кольори**. У області **Кольори** натискаємо 1ЛКМ на кнопці **Заливка** і вибираємо колір, яким і буде залитий створений прямокутник. Наприклад - коричневий. Клацаємо 1ЛКМ на кнопці **Лінії**, привласнюємо той же колір (коричневий), яким буде забарвлені лінії об'єкту.

Аналогічно створюємо другий прямокутник менших розмірів, відображаючий дно бункера, і помістимо його поряд з першим. Також побудуємо третій прямокутник, що є платформою для візка.

Майданчик готовий (рисунок 2).

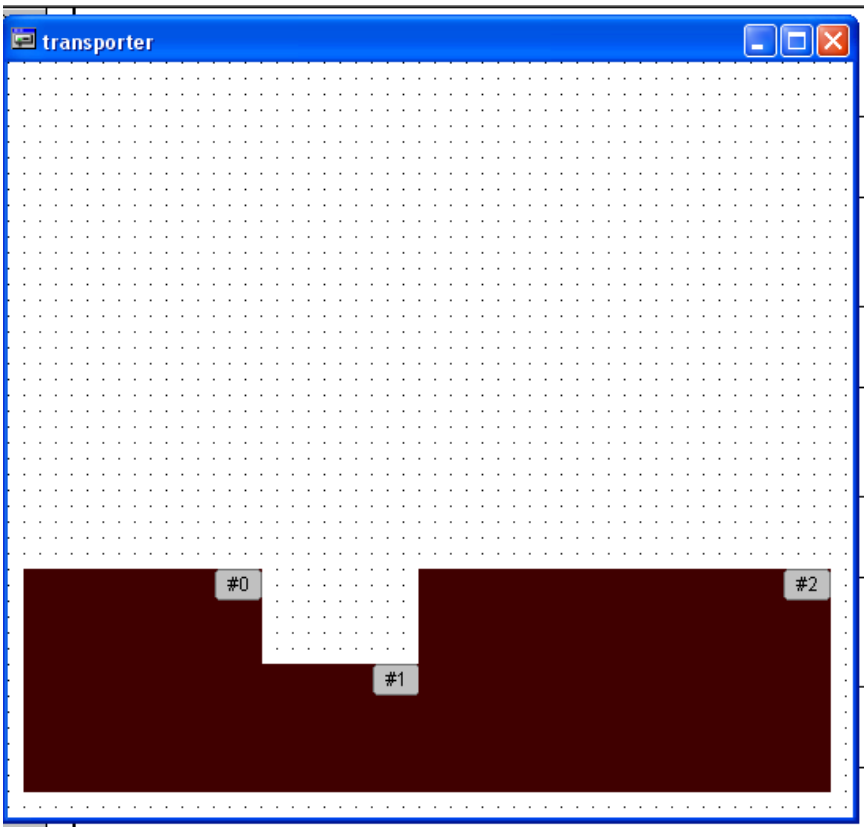

Рисунок 2 – Візуалізація майданчика лінії по транспортуванню сировини

Для візуалізації стану транспортерів виберемо найпростіший варіант індикації - зміну кольору.

Створимо графічну інтерпретацію транспортера **А** (рисунок 3).

Вибираємо елемент з меню Візуалізації **«Еліпс»** і нарисуємо коло з діаметром близько 1 сантиметра. Для цього клацнемо 1ЛКМ на робочому полі і, утримуючи ЛКМ, розтягнемо коло до необхідного розміру. Клацаємо 2ЛКМ

мишею на колі. З'явиться діалогове вікно для налаштування елементу візуалізації. Вибираємо категорію **Змінні** і в полі **Зм. кольору** вводимо ім'я змінної **PLC_PRG.M1**. Оскільки транспортер **А** наводиться в рух двигуном **М1**, то це ім'я зберігається і в програмі візуалізації з додаванням точки. Ця змінна буде управляти кольором намальованого кола.

Тепер виберемо кольори у спокої і в тривожному стані. Для цього вибираємо категорію **Кольори**. У області **Кольори** натисніть кнопку **Заливка** і у вікні, що з'явилося, натискаємо на будь-який нейтральний колір, наприклад, сірий. Натискаємо на кнопку **Заливка** в області **Тривожний колір** і вибираємо зелений колір. Отримане коло буде сірим, коли значення змінної неправдиве, і зеленою, коли змінна істинна. Копіюємо отримане коло і дублюємо воно.

Вставляємо прямокутник у візуалізацію між колами, для цього вибираємо в меню елемент «**Прямокутник».** Повторимо ті ж дії з введення змінної. Кольори виберемо ті ж : сірий і зелений. Перший транспортер закінчений.

Транспортер **В** (рисунок 3) розташований під кутом, тому процес створення його рисунка буде трохи складніший. Вибираємо в меню «Полігон», створюємо нахилений прямокутник. Техніка роботи з цим інструментом наступна: наводимо курсор миші на місце, з якого слід почати створення об'єкту. Клацаємо 1ЛКМ. Тепер відводимо курсор на місце наступної точки, натискаємо 1ЛКМ і так, проходячи усі точки, завершуємо побудову фігури клацанням 2ЛКМ на останній точці.

На кінцях прямокутника створюємо кола, тим самим отримуємо нахилений транспортер, вводимо у відповідність з вищеописаною методикою змінну **PLC_PRG.M2.** Не забуваємо вказати кольори (сірий і зелений). Обидва транспортери готові до роботи (рисунок 3).

Створення електродвигунів

Можна імпортувати растровий рисунок. Для цього вибираємо елемент «Растровий рисунок», виділяємо область додавання рисунку до необхідних розмірів за допомогою натиснутої ЛКМ. Після завершення операції розкривається діалог відкриття файлу. У теках комп'ютера знаходимо заздалегідь вибраний файл

з рисунком і завершуємо додавання кнопкою **Відкрити**. Після чого він з’явиться у виділеній області. Його розміри можна корегувати, а зображення переносити у вікні візуалізації, удержуючи його ЛКМ.

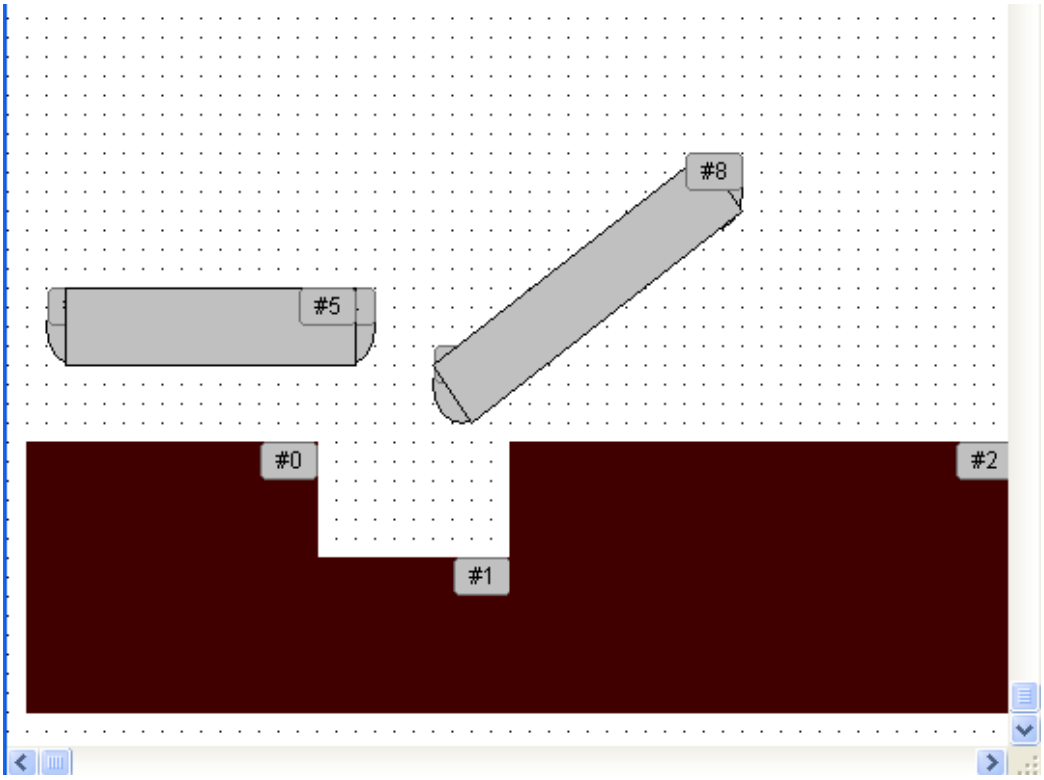

Рисунок 3 – Візуалізація транспортерів

Додати рисунки можна трьох форматів: *.bmp, *.jpg, *.tif. Не можна забувати, що вставлені графічні файли повинні знаходиться в теці з файлом програми, інакше вони зникнуть. Якщо ж немає готового зображення двигуна, то можна намалювати його за допомогою наявних елементів в меню **Візуалізації**.

Тепер додамо блок управління. Для цього вставляємо кнопку з меню візуалізації. Малюємо кнопку потрібних розмірів, натискаємо на ній 2ЛКМ. У полі **Рядок** категорії **Текст** вводимо ім'я кнопки **ПУСК** (рисунок 4).

Для того, щоб змінна **pusk** перемикалася при клацанні мишкою на цьому елементі, в полі **Змінна-кнопка** (рисунок 5) категорії **Введення** введемо змінну **PLC_PRG.PUSK** *не забуваємо крапку*). Створений перемикач включатиме транспортери.

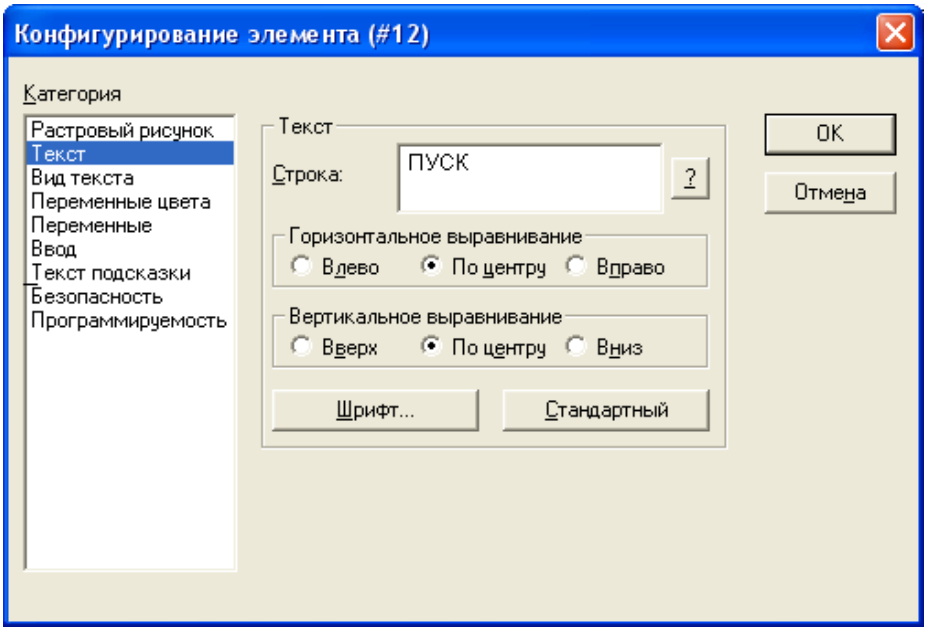

Рисунок 4 – Вікно ініціалізації кнопки

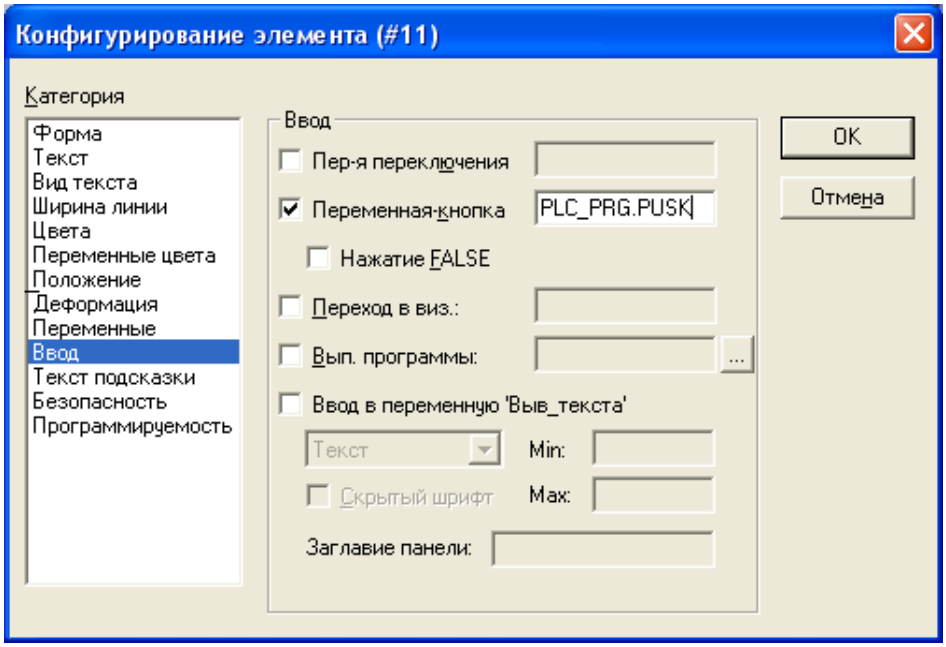

Рисунок 5 – Вікно ініціалізації кнопки

Ті ж дії виконаємо для управління кнопкою **СТОП**.

Тепер додамо останні штрихи у візуалізацію. Зображуватимемо візок за допомогою вже відомих елементів «Прямокутник», за допомогою якого ми зображуватимемо корпус візка і «Еліпс», який допоможе нарисувати колеса.

За допомогою елементу «Крива» додамо деякі елементи.

У результаті, візуалізація матиме вигляд, показаний на рисунку 6.

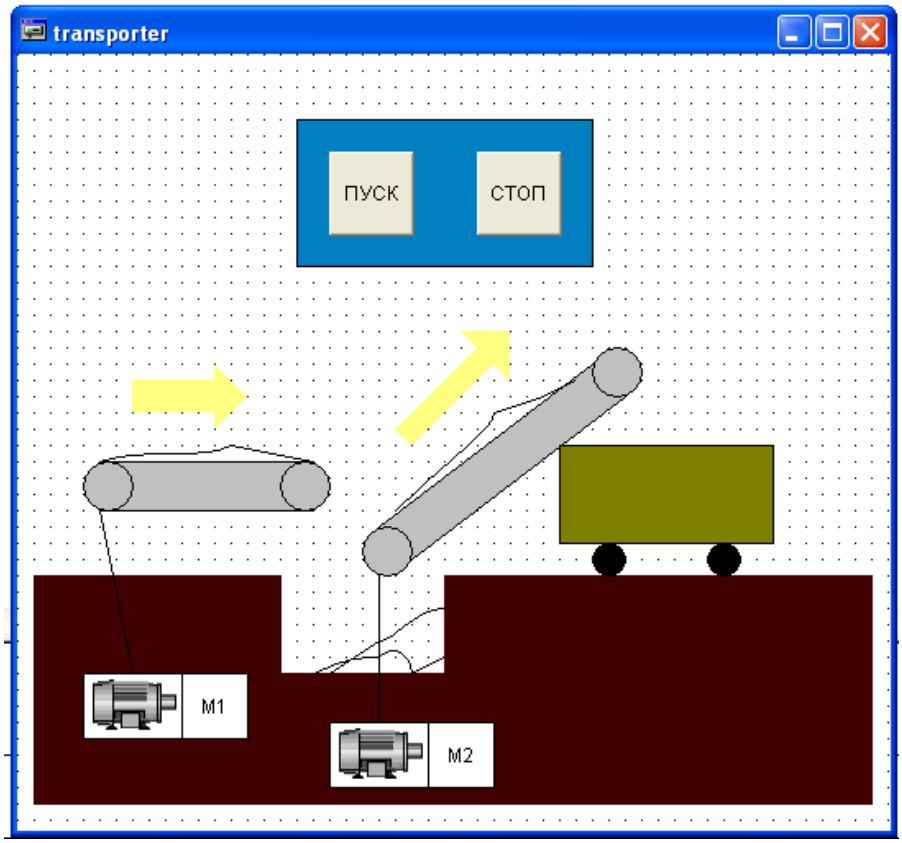

Рисунок 6 – Візуалізація транспортера

Проте статична картина на дисплеї, як відзначалося вище, притуплює пильність оператора. Тому краще ввести елементи з тривожною індикацією руху транспортерів за рахунок застосування ефекту мерехтіння їх забарвлення і/або додаткових стрілок, що вказують напрями переміщення сировини.

Для цього необхідно додати бібліотеку функціональних блоків. У вікні організаторів об'єктів виберемо вкладку **Ресурси** і в ній знайдемо **Менеджер бібліотек**, Вибираємо **Вставка - Додати бібліотеку**. Повинне відкритися діалогове вікно вибору файлів.

У вікні наявних бібліотек, що відкрилося, знаходимо **Util.lib**. Завершуємо операцію натисненням кнопки **Відкрити** (рисунок 7).

Познайомимося з новим компонентом **BLINK** (рисунок 8), який є генератором прямокутних сигналів. Для того, щоб включити його в ланцюг, користуємося ПКМ для додавання функціональних блоків. У вікні, що відкрилося, вибираємо нову бібліотеку, розділ **Signals** – **BLINK (FB)**. Генератор починає працювати за умови, що на його вхід **ENABLE** поступить сигнал TRUE,

після чого він починає видавати сигнали з виходу **OUT**. Параметри **TIMELOW** і **TIMEHIGH** типу **TIME**. Перший відповідає за час паузи, другий за тривалість імпульсу. Уставка значень по цих входах виконується по аналогії з параметром **PT** для таймерів.

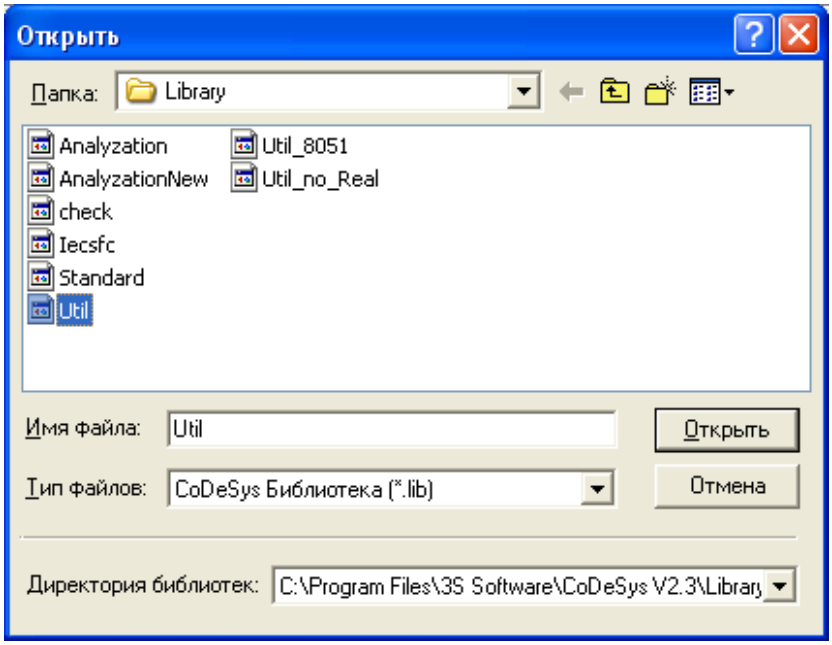

Рисунок 7 – Вибір нової бібліотеки

Застосування цього FB робить схему витонченішою (рисунок 8).

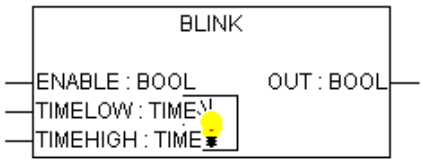

Рисунок 8 – Генератор прямокутних імпульсів BLINK

Внесемо зміни в нашу схему, для цього повернемося в програму (вікно організаторів проектів - **POU - PLC_PRG (PRG - LD)** і добавимо два ланцюги 9 і 10, як показано на рисунку 9, де також приведений повний список оголошених змінних.

Під час переходу входу **ENABLE** в стан **FALSE** значення виходу **OUT** визначається його станом у момент відключення входу. Так, якщо на виході була 1, вона і зберігається. Тому необхідно поставити замикаючі контакти **M1** і **M2** до і після **F11** і **F12** відповідно.

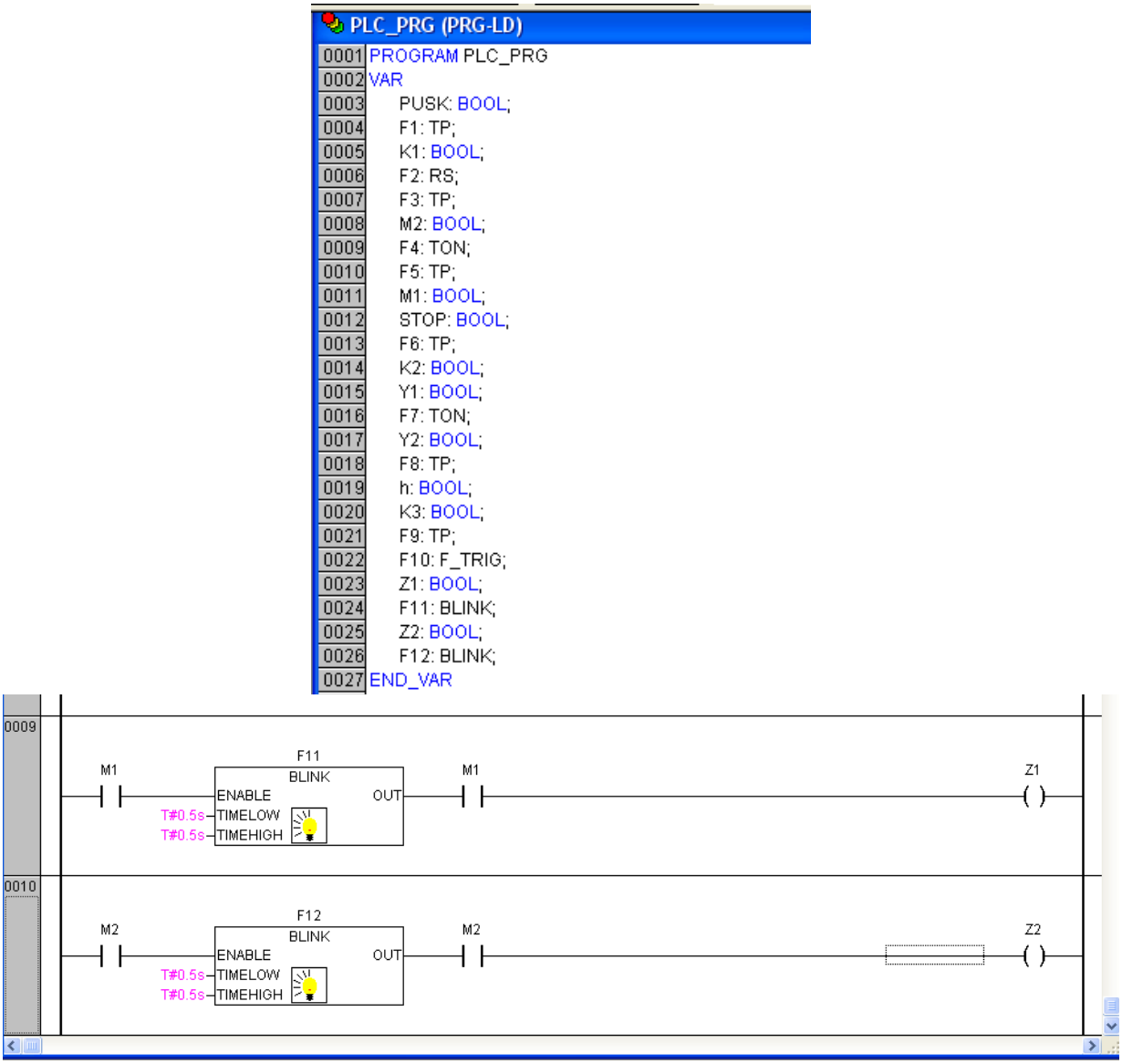

Рисунок 9 – Завершальні ланцюги програми

Можна ці контакти залишити тільки після цих **FB**. Тоді **F11** і **F12** постійно генеруватимуть імпульси. Але котушки **Z1** і **Z2**, як і планувалося, стануть «мерехтіти» тільки при включенні **M1** і/або **M2**.

Прокоментуємо створені ланцюги. Як вже відзначалося вище, змінна **M1** відповідає за роботу транспортера **A.** В ланцюзі 9 додаємо змінну **Z1**, яка відповідатиме за створення пульсацій стрілок на транспортері **A** при спрацьовуванні **M1**. Імпульси виробляються генератором **F11**. **Z1** є локальною змінною **VAR** для програми **PLC_PRG**. У ланцюг 10 ввели локальну змінну **Z2**, що відповідає за пульсацію стрілок на транспортері **B**, змінною якого є **M2.** Імпульси виробляються генератором сигналів **F12.**

Тепер повернемося у вікно, де створювали візуалізацію (Організатор об'єктів - Візуалізація - 2ПКМ на **transporter**).

За допомогою елементу «Полігон» над транспортерами створюємо стрілки, які сигналізуватимуть про роботу транспортерів. Вибираємо категорію **Кольори**. У області **Кольори Заливка** вибрати **нейтральну заливку**, поставити відмітку на **Лінії прозорий**. Натискаємо кнопку **Заливка** в області **Тривожний колір** і вибираємо, наприклад червоний колір. Задамо цим стрілкам значення змінних. Для стрілок, що відносяться до транспортера **A,** введемо змінну **PLC_PRG.Z1.** Для стрілок другого транспортера відповідно задаємо змінну **PLC_PRG.Z2**.

Кінцева візуалізація системи керування показана на рисунку 10.

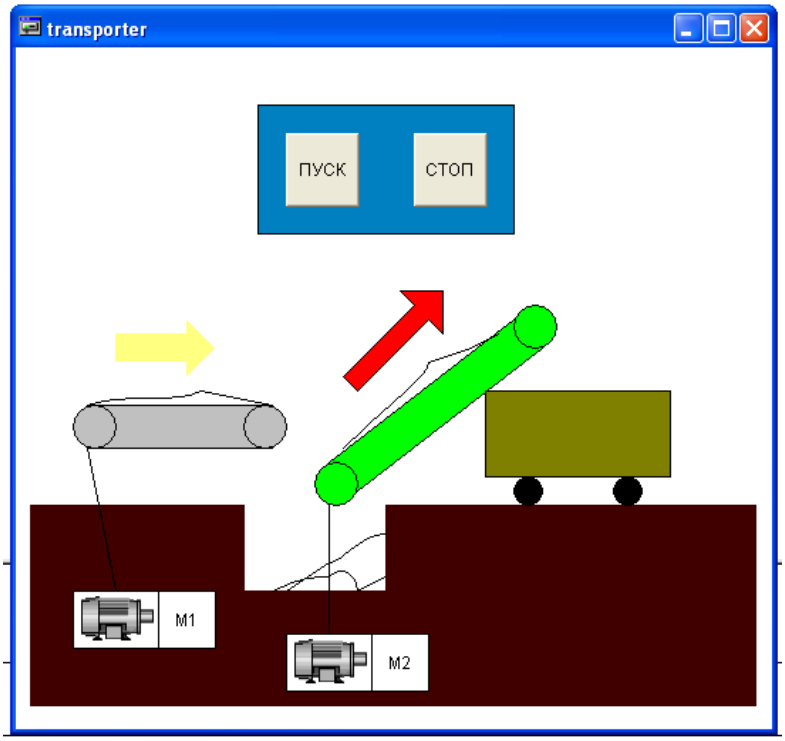

Рисунок 10 – Візуалізація системи керування

Запустити проєкт в режимі **емуляції**, **Онлайн -> Підключення, Старт**. Перейдіть у вікно візуалізації і перевірте роботу спроектованої системи керування.

Контрольні питання

1. Пояснить принцип роботи створеної системи керування.
2. Які елементи візуалізації були використані?
3. Як працює блок BLINK, до якої бібліотеки він належить?

ЛАБОРАТОРНА РОБОТА № 5

Тема: РОЗРОБКА СИСТЕМИ КОНТРОЛЮ РУХУ МЕХАНІЗМУ

Мета: отримати практичні навички проєктування ПЛК-програм контролю процесу оператором засобами мов FBD і SFC в середовищі CoDeSys v2.3.

Постановка задачі. Створити ПЛК-програму для контролю руху механізму.

Короткі теоретичні відомості

Мова FBD (функціональних блокових діаграм)

FBD - це графічна мова програмування. У редакторах LD і FBD програма представлена у вигляді списку ланцюгів. Кожен ланцюг складається з двох частин: в лівій записаний номер ланцюга, а в правій - структура, що складається з логічних або арифметичних операцій, викликів програм, функцій або функціональних блоків, інструкцій переходу або повернення.

Найбільш важливі функції можна знайти в контекстному меню, натиснувши ПКМ.

Команди створення ланцюгів

Якщо ви хочете вставити новий ланцюг в редакторові FBD, то використайте команди меню **Вставити**: **Ланцюг (після)** і **Ланцюг (перед)** для вставки ланцюга після або перед вибраним ланцюгом відповідно. Щоб вибрати схему, клацніть мишею на потрібному вам ланцюзі. Номер поточного ланцюга виділяється прямокутником з пунктирною межею. Якщо натиснути <Shift>, те можна виділити відразу декілька схем, клацаючи мишею по кожній.

Швидке введення команди **Ланцюг (після)** : **<Shift> +<T>**.

Основні команди редактора FBD

Основні команди графічного редактора мови FBD можна виконати з допомогою:

- меню Вставити системи CoDeSys;

- панелі інструментів редактора FBD (рисунок 1);

- контекстного меню (рисунок 2).

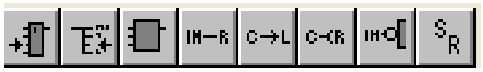

Рисунок 1 – Панель інструментів редактора FBD

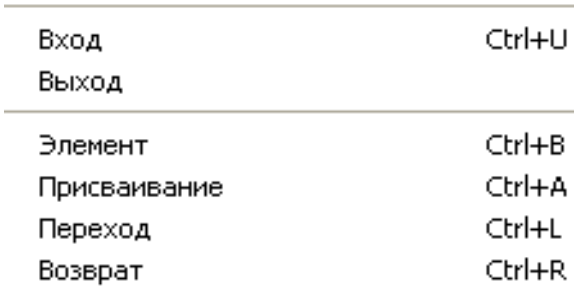

Рисунок 2 – Елементи контекстного меню

Команди редагування FBD- діаграми

Ці команди можна знайти в меню **Правка** (контекстне меню) (рисунок 3):

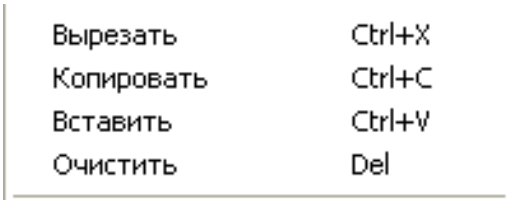

Рисунок 3 – Команди меню Правка

Мова SFC (послідовних функціональних схем)

SFC - це графічна мова, яка дозволяє описати хронологічну послідовність різних дій в програмі. Для цієї дії зв'язуються з кроками (етапами), а послідовність роботи визначається умовами переходів між кроками.

Крок

SFC POU складається з набору кроків, пов'язаних переходами. Існують 2 види кроків :

• Крок простого типу (спрощений SFC) може включати єдину дію. Графічний прапорець (невеликий трикутник у верхньому кутку кроку) показує, порожній крок або ні.

• МЭК крок (стандартний SFC) пов'язаний з довільним числом дій або логічних змінних. Пов'язані дії розташовуються з правого боку від кроку.

Дія

Дія може утримувати список інструкцій на IL або ST, схеми на FBD або LD, або знову схеми на SFC. Окрім основної дії, крок може включати одне вхідне і одна вихідна дія.

Дії МЭК кроків показані в Організаторові Об'єктів, безпосередньо під тією, що викликає їх POU. Редагування дії запускається подвійним клацанням миші або клавішею <Enter>. Нові дії додаються командою головного меню 'Проект' 'Додати дію' ('Project' 'Add Action'). Ви можете зіставити одному кроку до 9 дій.

При використанні простих кроків дія завжди зв'язується з цим кроком. Для того, щоб редагувати дію, необхідно двічі клацнути лівою клавішею мишки на кроці. Чи виділити крок і вибрати команду меню 'Доповнення' 'Відкрити дію/Перехід' ('Extras' 'Zoom Action/Transition').

Перехід/умова переходу

Між кроками знаходяться так звані переходи. Умовою переходу може бути логічна змінна або константа, логічна адреса або логічний вираз, описаний на будь-якій мові. Умова може включати декілька інструкцій, що утворюють логічний результат, у вигляді ST вираження (тобто (i<=100) AND b) або на будь-якій іншій мові. Але умова не повинна містити привласнення, виклик програм і екземплярів функціональних блоків!

У редакторові SFC умову переходу можна записати безпосередньо біля символу переходу або в окремому вікні редактора для введення умови ('Доповнення' 'Відкрити дію/Перехід' ('Extras' 'Zoom Action/Transition')). Умова, задана у вікні редактора переважніше!

Активний крок

Після виклику SFC POU початковий крок (крок, виділений подвійною рамкою) виконується першим. Крок, що виконується в даний момент, називається активним. Дії, пов'язані з активним кроком, виконуються один раз в кожному

циклі, що управляє. У режимі Онлайн активні кроки виділяються синім кольором. Крок, що йде за активним кроком, стане активним, тільки коли умова переходу до цього кроку прикмет значення TRUE.

У кожному циклі, що управляє, будуть виконані дії, що містяться в активних кроках. Далі перевіряються умови переходу, і, можливо, вже інші кроки стають активними, але виконуватися вони будуть вже в наступному циклі.

Альтернативна гілка

Дві і більше гілки SFC можуть бути альтернативними. Кожна альтернативна гілка повинна починатися і закінчуватися переходом. Альтернативні гілки можуть містити паралельні гілки і інші альтернативні гілки. Альтернативна гілка починається горизонтальною лінією (початок альтернативи), а закінчується горизонтальною лінією або переходом на довільний крок (jump). Якщо крок, який знаходиться перед лінією альтернативного початку, активний, то перші переходи альтернативних гілок починають оцінюватися зліва направо.

Таким чином, першим активується той крок, який йде за першим ліворуч істинним переходом.

Паралельні гілки

Дві і більше гілки SFC можуть бути паралельними. Кожна паралельна гілка повинна починатися і закінчуватися кроком. Паралельні гілки можуть містити альтернативні гілки і інші паралельні гілки. Паралельна гілка наноситься подвійною горизонтальною лінією і закінчується подвійною горизонтальною лінією (кінець паралелі) або переходом на довільний крок (jump).

Якщо крок активний, умова переходу після цього кроку істинна і за цим переходом йдуть паралельні гілки, то активуються перші кроки цих гілок. Ці гілки виконуються паралельно один одному. Крок, що знаходиться після паралельних гілок, стає активним тільки тоді, коли усі попередні кроки активні і умова переходу істинна.

Перехід на довільний крок (Jump)

Перехід на довільний крок - це з'єднання на крок, ім'я якого вказане під знаком «jump». Такі переходи потрібні для того, щоб уникнути пересічних з'єднань, що йдуть вгору.

Порядок виконання роботи

Постановка задачі. Створити ПЛК-програму для контролю руху механізму. Оператор повинен періодично підтверджувати правильність функціонування механізму. Інакше, необхідно видати попередження, а потім зупинити роботу. Робочий орган механізму здійснює циклічний рух по периметру прямокутника.

Рішення задачі

Запустіть CoDeSys v2.3 (ярлик  на Робочому столі) і виберіть '**Файл**' - '**Створити**'.

Налаштування цільової платформи

При створенні проєкту вибираємо цільову конфігурацію **3SCoDeSys SP PLCWint V2.4.**

Головна програма PLC_PRG POU

Наступне діалогове вікно визначає тип першого програмного компонента (**POU**). Виберете мову реалізації **FBD** і збережете запропоновані за умовчанням тип компонента - **Програма** і ім'я - **PLC_PRG**.

Оголошення перемикача підтвердження

Перемикач підтвердження - це змінна, яка змінюватиме значення при підтвердженні коректності роботи механізму оператором.

У першому ланцюзі графічного FBD редактора виділіть рядок питань ??? і введіть найменування першої змінної. Нехай це буде **Observer (**спостерігач). Тепер натисніть на клавіатурі Enter. У діалозі оголошення змінної, що з'явився, збережете ім'я **Observer** і логічний тип **BOOL**. Змініть клас змінної на глобальний **VAR_GLOBAL**. Підтвердіть визначення - **ОК.** Тепер визначення змінної

Observer повинне з'явитися у вікні глобальних змінних проєкту **Global_Variables** на вкладці **Ресурси** Організатора об'єктів (рисунок 4).

Рисунок 4 – Оголошення глобальної змінної

Підключення standard.lib

Необхідно підключити стандартну бібліотеку. Для цього відкрийте менеджер бібліотек командами **Вікно - Менеджер бібліотек**. Виберіть **Вставка - Додати бібліотеку**. Повинне відкритися діалогове вікно вибору файлів. Виберіть **STANDARD.lib** зі списку бібліотек (рисунок 5).

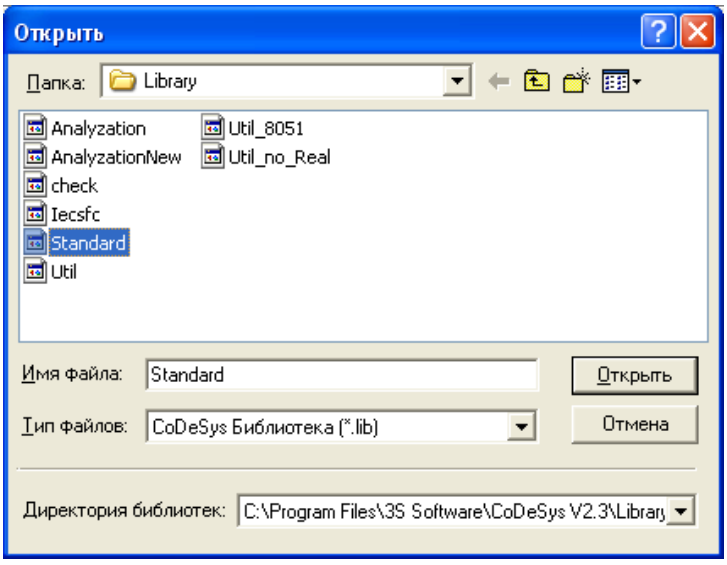

Рисунок 5 – Підключення стандартної бібліотеки

Детектор переднього фронту

Оператор повинен підтверджувати роботу саме перемиканням клавіші. Для цього необхідно визначити моменти натиснення і відпуску, тобто переходи значення логічної змінної з нуля (FALSE) в одиницю (TRUE) і навпаки.

У вікно редактора PLC_PRG виділите позицію праворуч від змінної **Observer**. З'явиться маленький пунктирний прямокутник. Клацніть по ньому правою клавішею миші. У контекстному меню введення задайте команду **Елемент**.

За умовчанням, вставляється елемент **AND**. Виділіть ім'я **AND,** натисніть клавішу **F2** (асистент введення). У діалоговому вікні(ліворуч) виберете категорію: стандартні функціональні блоки. З тригерів (**Trigger**) стандартної бібліотеки виберете **R_TRIG**. R_TRIG формує логічну одиницю по передньому фронту на вході (рисунок 6).

Необхідно задати ім'я для нового екземпляра функціонального блоку R_TRIG. Клацніть мишкою по ??? над зображенням тригера і введіть ім'я **Trigl**. У діалозі визначення змінних має бути вказаний клас **VAR** (локальні змінні), ім'я **Trigl** і тип **R_TRIG**. Натисніть **ОК (**рисунок 7).

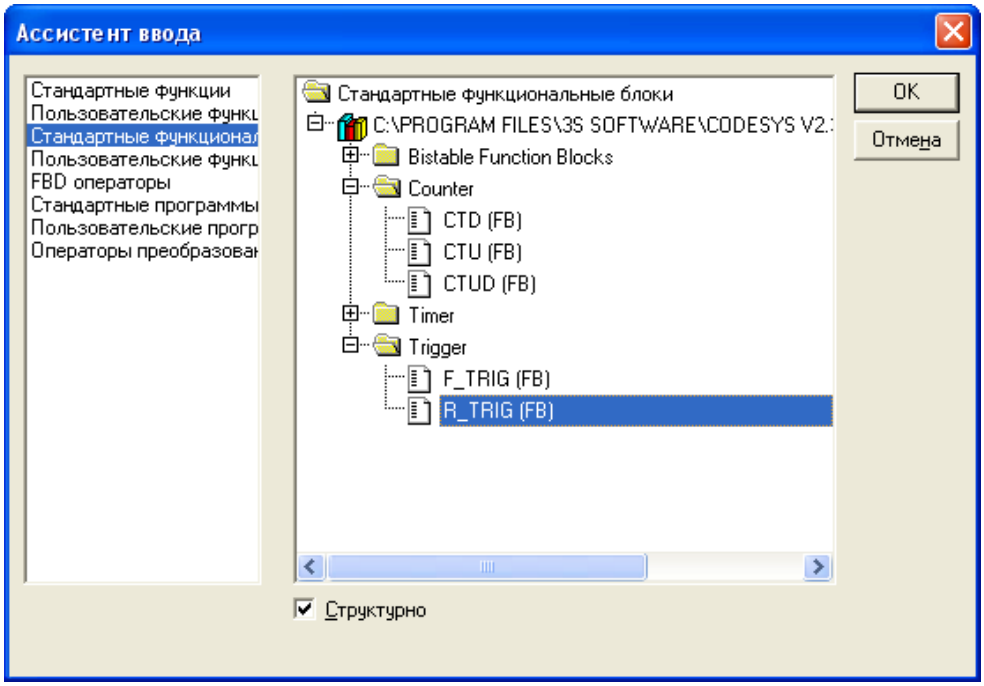

Рисунок 6 – Вибір тригера

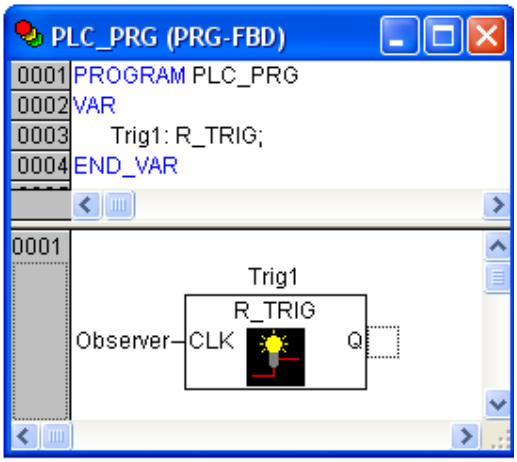

Рисунок 7 – Оголошення програми PLC_PRG

Детектор заднього фронту

Виділіть вихід функціонального блоку **Trigl** і вставте (як було описано вище) елемент **AND** і перейменуйте його в **OR (**логічне АБО). Виділіть вільний вхід функціонального блоку **OR** і вставте перед ним екземпляр функціонального блоку F_TRIG під ім'ям Trig2. На вхід F_TRIG, за допомогою асистента введення (F2) подайте (категорія **Глобальні змінні**) змінну **Observer** (рисунок 8).

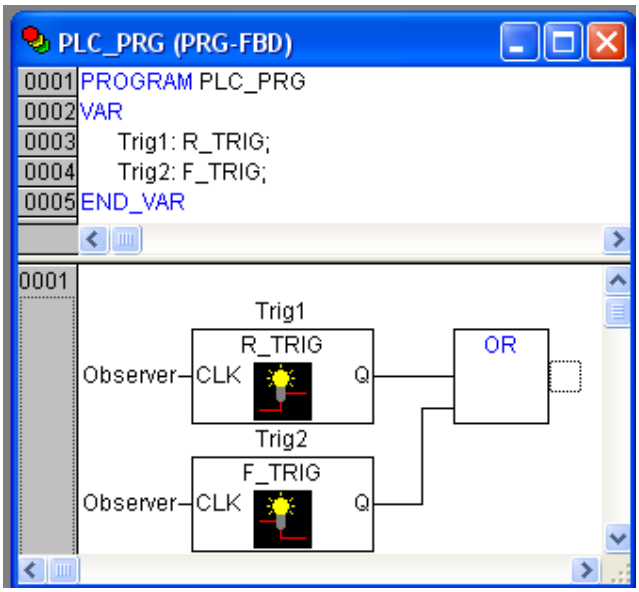

Рисунок 8 – Фрагмент програми PLC_PRG

Контроль часу

Вставте після OR екземпляр функціонального блоку **TOF** (таймер із затримкою виключення із стандартної бібліотеки) під ім'ям **Timerl**. Замініть ??? на вході **PT** константою **T#10s** (*взяти з таблиці для свого варіанту*). Вона відповідає 10 секундам.

Вихід Попередження

Виділіть вихід **Q** таймера **Timerl** і в контекстному меню дайте команду Присвоювання. Замініть питання на ім'я змінної Warning. У діалозі визначення задайте їй клас VAR_GLOBAL і тип BOOL.

Тепер виділите позицію в середині лінії тій, що з'єднує вихід таймера і змінну **Warning**. Задайте команду **Інверсія** в контекстному меню. Маленьке коло означає інверсію значення логічного сигналу (рисунок 9).

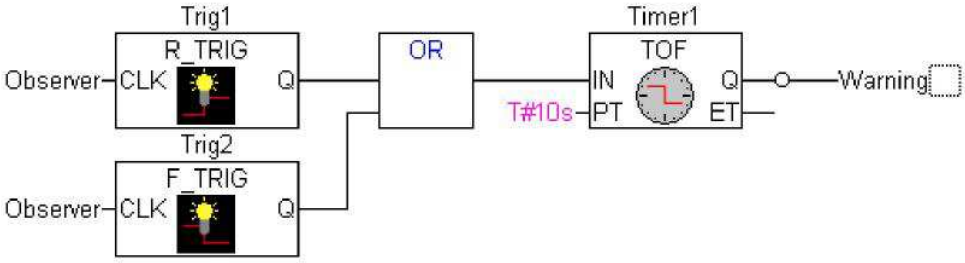

Рисунок 9 – Перший ланцюг програми PLC_PRG

Формуємо Стоп Сигнал по другому інтервалу часу

Створіть новий ланцюг командою меню **Вставити→Ланцюг (після)**. Вставте із стандартної бібліотеки в новий ланцюг **Елемент** типу **TON(**таймер із затримкою включення) під ім'ям **Timer2**. Подайте змінну **Warning** на вхід **IN** (використайте асистент введення **F2)** і константу **T** (*узяти з таблиці. 3.1.1 для свого варіанту*) на вхід **PT.** Вихід екземпляра функціонального блоку **Timer2** присвойте **(Присвоювання)** новій глобальній (Клас **VAR_GLOBAL)** логічній змінній **Stop** (рисунок 10).

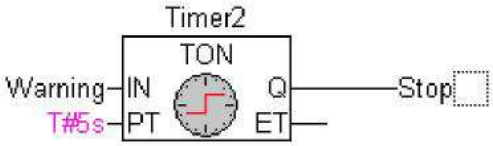

Рисунок 10 – Другий ланцюг програми PLC_PRG

POU управління механізмом

У організаторові об'єктів створити новий POU з ім'ям **Mashine**, типом **Програма** і визначити для нього мову **SFC**.

За умовчанням, створюється порожня діаграма, що містить початковий крок "Init" і відповідний перехід "Trans0", який закінчується поверненням до Init (рисунок 11). Ми далі використовуватимемо спрощений SFC, без МЕК дій.

Визначаємо послідовність роботи механізму

Кожній фазі роботи повинен відповідати певний етап (крок). Виділіть перехід (Trans0) так, щоб він виявився оточений пунктирною рамкою. У контекстному меню дайте команду вставки кроку і переходу під виділеним: **Крок-перехід (знизу**). Аналогічно повторите вставку ще 4 рази. Включаючи Init, повинне вийти 6 кроків з переходами.

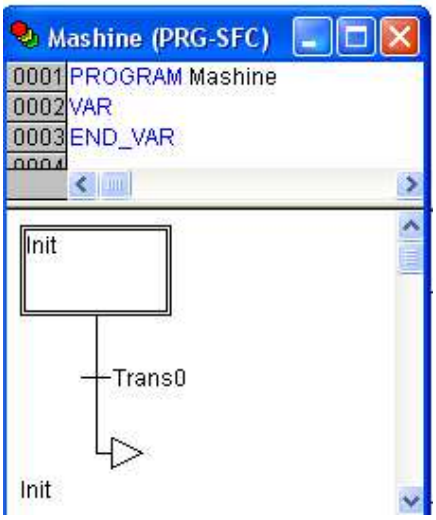

Рисунок 11 – Діаграма SFC

Клацаючи мишею по іменах переходів і кроків, можна помітить, що вони виділяються кольором. У такий спосіб можна визначити нові найменування.

Перший після **Init** крок повинен назватися **Go_Right**. Під ним **Go_Down**, **Go_Left**, **Go_Up** і **Count**.

Програмування першого кроку

Клацніть двічі на кроці **Go_Right**. CoDeSys почне визначення дії кроку і попросить вибрати мову його реалізації. Виберете **ST** (structured text) і перейдіть в автоматично відкрите вікно текстового редактора. У цьому кроці робочий орган нашого механізму повинен переміщатися по осі X управо. Програма повинна виглядати так:

X_pos:= X_pos + 1;

Завершіть введення клавішею Enter, і визначите змінну **X_pos** типу **INT**.

Програмування наступних кроків

Повторіть описану послідовність для усіх кроків, що залишилися. Змінні **Y_pos** і **Counter** мають бути типу **INT**.

Крок **Go_Down** програма **Y_pos:= Y_pos + 1;**

Крок **Go_Left** програма **X_pos:= X_pos - 1;**

Крок **Go_Up** програма **Y_pos:= Y_pos - 1;**

Крок **Count** програма **Counter := Counter + 1;**

Визначення переходів

Перехід повинен містити умову, що дозволяє перемикання на наступний крок. Перехід після кроку Init назвіть **Start** і визначите нову логічну змінну (Клас **VAR_GLOBAL** тип **BOOL**). При одиничному значенні цієї змінної починається цикл роботи механізму.

Наступний перехід повинен містити умову **X_Pos = 100** (*узяти з таблиці 1 для свого варіанту*), так при значенні позиції X включається наступна фаза руху. Умова третього кроку **Y_pos = 50** (*узяти з таблиці 1 для свого варіанту*), четвертого **X_pos = 0**, п'ятого **Y_pos = 0** і шостого **TRUE** (рисунок 12).

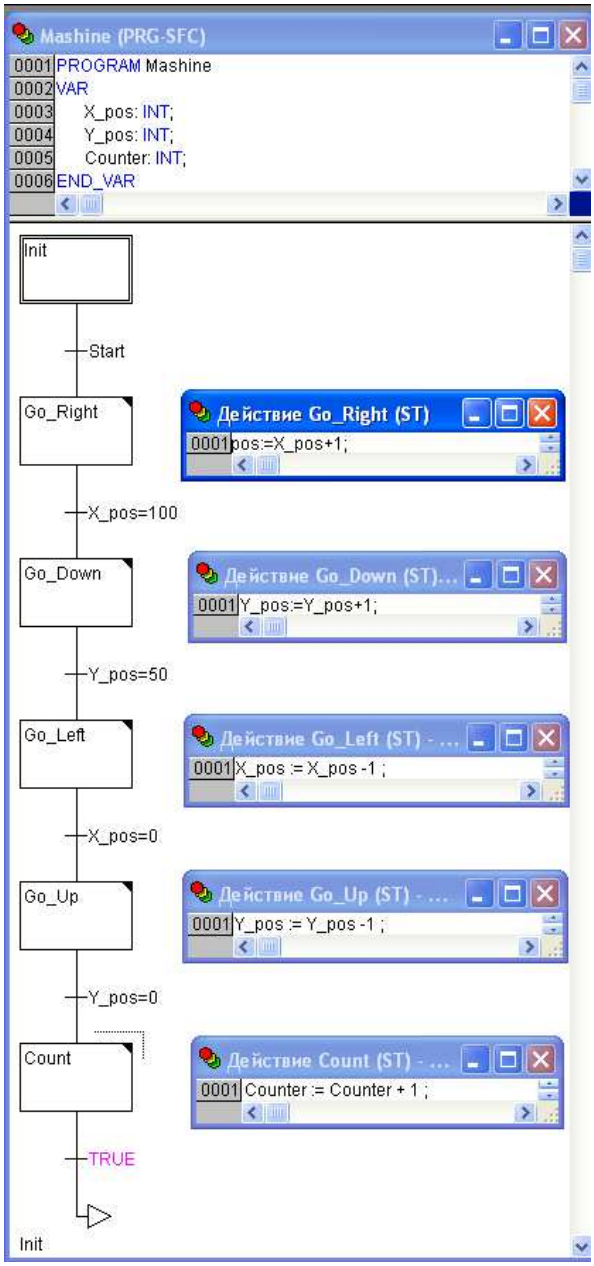

Рисунок 12 – Програма Mashine

Зупинення механізму

Поверніться до POU **PLC_PRG** і додайте третій ланцюг.

Замість питань вставте змінну **Stop**, і потім з контекстного меню вставте оператор **Повернення**. Return перериває роботу програми PLC_PRG POU при одиничному значенні Stop (рисунок 13).

Рисунок 13 – Третій ланцюг програми PLC_PRG

Виклик POU управління механізмом

Додайте ще один ланцюг, виділіть його і вставте **Елемент** з контекстного меню. Як завжди це буде "AND". Натисніть <F2> і в асистентові введення задайте POU управління механізмом в категорії **Призначені для користувача програми** (рисунок 14).

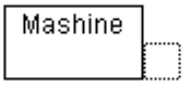

Рисунок 14 – Четвертий ланцюг програми PLC_PRG

Компіляція проєкту

Відкомпілюйте проект цілком. Якщо ви усі зробили вірно, то в нижній частині вікна повинне з'явитися повідомлення: «**0 помилок»**. Інакше необхідно виправити допущені помилки. У це допоможуть розгорнуті повідомлення про помилки.

Створення візуалізації

Перейдіть на вкладку візуалізації. У контекстному меню введіть команду додавання об'єкту. Присвойте новому об'єкту ім'я **Observation**.

У кінці роботи, вікно візуалізації виглядатиме як показано на рисунку 15.

Елемент візуалізації

Почнемо малювати перемикач підтвердження (на рисунку 15 прямокутник з текстом **ОК**).

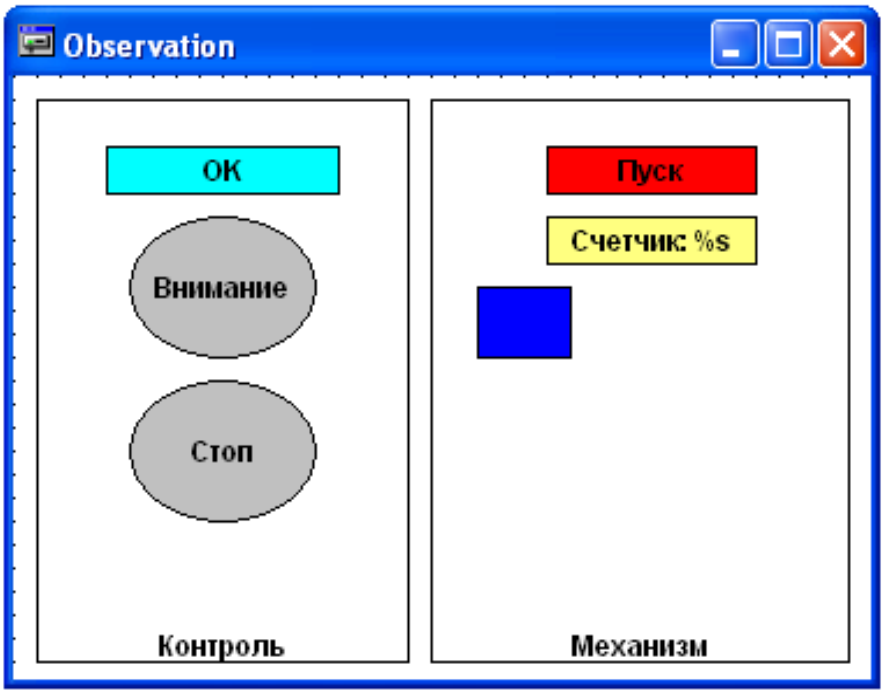

Рисунок 15 – Вікно візуалізації

На панелі інструментів виберіть елемент **Прямокутник**. У вікні редактора візуалізації натисніть ліву клавішу миші і розтягніть прямокутник до потрібної висоти і ширини, відпустіть клавішу.

Налаштування першого елементу візуалізації

Діалогове вікно налаштування елементу викликається подвійним клацанням миші на його зображенні. Задайте у віконці **Рядок** категорії **Текст** слово **ОК**.

Тепер перейдіть в категорію **Змінні**, клацніть мишею в полі зміна кольору і скористайтеся асистентом введення **F2**. Вставте змінну **.Observer** зі списку глобальних змінних. Далі перейдіть в категорію **Кольори**. Задайте колір зафарбовування елементу (**Заливка**), наприклад, ясно-блакитний. Для «тривожного» стану необхідно визначити інший колір (**Тривожний колір**), наприклад, блакитний. У категорії **Введення** необхідно ще раз ввести змінну **Observer** і поставити прапорець **Зм-а перемикання**. Закрийте діалог налаштування.

У результаті, прямокутник відображатиметься ясно-блакитним при значенні змінної **Observer** рівному FALSE і блакитним, при значенні TRUE. Її значення змінюватиметься при кожному «натисненні» нашої клавіші.

Розвиток візуалізації

Намалюйте коло **Увага** з наступними налаштуваннями:

Категорія **Текст**, поле **Рядок:** текст **Увага**.

Категорія **Змінні**, поле **Зм. Кольору:** змінна **.Warning**

Категорія **Кольори**, кнопка **Заливка** сірим кольором, **Тривожний колір** червоним кольором.

Скопіюйте створене коло командою **Правка → Копіювати** і вставте її один раз командою **Правка → Вставити** для нового кола **Стоп** нижче кола **Увага**.

Виправіть налаштування нового кола **Стоп**:

- Категорія **Текст**, поле **Рядок:** текст **Стоп**.
- Категорія **Змінні**, поле **Зм. Кольору:** змінна **.Stop**

Намалюйте прямокутник для клавіші **Пуск**, який має наступні налаштування:

- Категорія **Текст**, поле **Рядок:** текст **Пуск**
- Категорія **Змінні**, поле **Зм. Кольору:** змінна **.Start**
- Категорія **Введення**, прапорець **Зм-а перемикання** включений, змінна **.Start**
- Категорія **Кольори**, кнопка **Заливка** - червоним, і **Тривожний колір** - зеленим.

Намалюйте прямокутник для лічильника з наступними налаштуваннями:

- Категорія **Текст**, поле **Рядок:** текст **Лічильник: %s** (**%**s заступник для відображення значення змінної).
- Категорія **Змінні,** поле **Вив_тексту :** змінна **Mashine.Counter**.
- Категорія **Кольори**, кнопка **Заливка** - жовтим кольором.

Намалюйте невеликий прямокутник, що означає робочий інструмент механізму, з наступними налаштуваннями:

- Категорія **Положення,** поле **Зрушення по X:** змінна **Mashine.X_pos**; поле **Зрушення по Y:** змінна **Mashine.Y_pos**.
- Категорія **Кольори**, кнопка **Заливка** - синім кольором.

Намалюйте дві декоративні рамки для розділення областей контролю і механізму. Задайте в них відповідні написи з вирівнюванням по низу (**Вертикальне вирівнювання**). Використовуючи контекстне меню, помістите декоративні прямокутники на задній план.

Запуск проєкту в режимі емуляції

Команда запускає **Онлайн -> Підключення Старт** проект. Перейдіть у вікно візуалізації і перевірте роботу механізму. Зафіксуйте різні положення механізму.

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№ вар.	Час PT Timer1, сек	Час PT Timer2, сек	Крок по X	Крок по Y
1	16	8	150	60
2	20	10	200	100
3	8	4	120	70
4	12	6	160	90
5	16	10	130	80
6	14	8	140	70
7	18	12	100	60
8	10	6	110	60
9	14	7	160	80
10	18	9	170	80
11	20	12	110	50
12	14	10	140	80
13	12	8	170	70

14	10	6	180	80
15	20	14	100	50
16	8	6	190	90
17	22	11	180	60
18	24	12	190	80
19	18	10	130	60
20	10	5	150	70

Контрольні питання

1. Мова FBD і структура FBD-ланцюга.
2. Як створюються ланцюги?
2. Які основні елементи мови FBD?
3. Які основні команди редагування редактора FBD?
4. Мова SFC.
5. Кроки.
6. Переходи.
7. Початковий крок.
8. Активний крок.
9. Дії.
10. Програмування кроків і переходів.
11. Паралельні гілки.
12. Альтернативні гілки.
13. Перехід на довільний крок.
14. Виділення блоків в SFC.
15. Який блок генерує імпульс за заднім фронтом вхідного сигналу?
16. Який блок генерує імпульс за переднім фронтом вхідного сигналу?

ЛАБОРАТОРНА РОБОТА № 6

Тема: РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ СВІТЛОФОРОМ

Мета роботи: отримати практичні навички проєктування ПЛК-програм управління за часом процесами засобами мов стандарту МЭК 61131-3 в середовищі CoDeSys v2.3.9; познайомитися із засобами моделювання в CoDeSys.

Постановка задачі: створити ПЛК-програму для управління світлофором.

Короткі теоретичні відомості

Мова лінійних інструкцій IL

Текст програми на Il – список послідовних інструкцій. Кожен запис в окремому рядку. Інструкція може містити 4 поля, розділені пробілом або горизонтальної табуляцією:

Мітка: Оператор Операнд Коментар

Мітка не є обов'язковою. Ставиться тільки там, де потрібна. Оператор присутній обов'язково, операнд - за потребою. Коментар - не обов'язкове поле, ставиться в кінці рядка. Ставити коментар між полями інструкцій не можна. Для кращого сприйняття колонки полів вирівнюють.

Редактор CoDeSys вирівнює текст автоматично. Крім того він виконує синтаксичний контроль і виділення кольором помилки (некоректно введений оператор виділяється блакитним кольором).

Акумулятор

Абсолютна більшість операторів IL виконує деяку інструкцію з акумулятором. Операнд теж може брати участь в інструкції, але результат розміщується в аккумулятор.

Наприклад, інструкція **add 2** додає до вмісту оператора число 2 і поміщає результат в акумулятор. Команди порівняння порівнюють вміст операнда і акумулятора, результат порівняння ІСТИНА або ХИБНІСТЬ поміщається в акумулятор. Команди переходу на мітку здатні аналізувати вміст акумулятора і приймати рішення - виконувати перехід чи ні. Таким чином, акумулятор IL є

універсальним, здатним зберігати значення змінних будь-якого типу. В акумулятор можна помістити значення типу Bool потім Int або Real - транслятор помилки не видасть. Така гнучкість не означає, що акумулятор може одночасно містити кілька значень різних типів. Він містить тільки одне значення, причому тип значення фіксується в акумуляторі, і, якщо операція зажадає значення іншого типу транслятор видасть помилку.

Перехід на мітку

Програма на IL виконується підряд зверху вниз. Для зміни порядку виконання використовується оператор JMP. Цей оператор виконується завжди незалежно ні від чого. Оператор JMPC виконується тільки при значенні акумулятора - ІСТИНА. Перехід можна виконувати як вверх так і вниз. Мітки тільки локальні (тобто перехід на мітку в іншому POU неможливий). Переходи потрібно організовувати акуратно, щоб не отримати безкінечний цикл.

Приклад:

```
        LD    1
M2:     ADD   1
        ST    Y
        LE    5
        JMPC  M2
```

В даному прикладі передбачалася організація циклу на 5 повторень.

У CoDeSys команда безумовного переходу виконується дуже швидко, тому що транслюється в одну машинну команду. Обмеження на число переходів немає.

```
LD    2
ADD   1
ST    Y
JMP   M2
```

Виклик функіонального блоку

Викликати екземпляр FB або програму в IL можна з одночасним присвоюванням змінних.

Приклад:

```
CAL    CTD(CD:= TRUE, LOAD:=FALSE, PV:=100)
LD     CTD.CV
ST     Y
```

Аналогічний виклик можна виконати з попереднім присвоюванням значень вхідних змінних.

```
LD     TRUE
ST     CTD.CD
LD     FALSE
ST     LOAD
LD     100
ST     PV
CAL    CTD
LD     CTD.CV
ST     Y
```

Виклик функції

При виклику функції перерахування параметрів в IL присутня одна чимало важлива особливість - в якості першого параметра використовується акумулятор.

Приклад:

```
LD TRUE
SEL 3,4
```

Мовою ST це рівносильно виклику SEL (TRUE, 3,4). Очевидно, при виклику функції або оператора з одним параметром список параметрів взагалі не потрібен

Модифікатори

Додавання до мнемоніки деяких операторів символів - модифікаторів '**C**' і '**N**' модифікує зміст інструкції.

Символ '**N**' (negation) викликає інверсію значення операнда до виконання інструкції. Операнд повинен бути типів **BOOL**, **BYTE**, **WORD** або **DWORD**.

Символ '**C**' (condition) додає перевірку умов до командам переходу, виклику і повернення. Команди **JMPC**, **CALC**, **RETC** будуть виконуватися тільки при значенні акумулятора ІСТИНА. Додавання символу '**N**' призводить до порівняння умови з інверсним значенням акумулятора. Команди **JMPCN**, **CALCN**, **RETCN** будуть виконуватися тільки при значенні акумулятора ХИБНІСТЬ. Модифікатор '**N**' без '**C**' не має сенсу в даних операціях і не застосовується.

Оператори IL

Стандартні оператори IL з допустимими модифікаторами представлені в таблиці 1.

Таблиця 1 - Стандартні оператори мови IL

Оператор	**Модифікатор**	**Опис**
LD	N	Завантажити значення операнда в акумулятор
ST	N	Присвоїти значення акумулятора операнду
S		Якщо акумулятор TRUE, установити логічний операнд на TRUE
R		Якщо акумулятор TRUE, скинути логічний операнд на FALSE
AND	N, (	Логічне І
OR	N, (	Логічне АБО
XOR	N, (	Поразрядне АБО
NOT		Поразрядна інверсія акумулятора
ADD	(	Складання
SUB	(	Віднімання
MUL	(	Множення
DIV	(	Ділення
MOD	(	Ділення по модулю
GT	(	Перевірити: >

GE	(	Перевірити: >=
EQ	(	Перевірити: =
LE	(	Перевірити: <=
LT	(	Перевірити: <
NE	(	Перевірити: < >
CAL	C N	Виклик функціонального блоку
JMP	C N	Перехід на мітку
RET	C N	Вихід з POU, повернення до програми

Порядок виконання роботи

Постановка задачі. Створити ПЛК-програму для блоку управління рухом на перехресті, що має світлофори для двох пересічних напрямів руху. Світлофори повинні мати два протилежні стани - червоний і зелений, а також перехідні стадії: жовтий і жовто-червоний, такий, що включається перед зеленим. Остання стадія має бути довша за попередню. Управління світлофором повинно здійснюватися за допомогою кнопки. Час роботи сигналів і інші параметри роботи світлофорів вибираються з таблиці 1 відповідно до варіанту.

Рішення задачі

Конфігурація проєкту

Запустіть CoDeSys v2.3 і виберіть '**Файл**' - '**Створити'.** З'явиться діалогове вікно налаштування цільової платформи, вибираєте **3SCoDeSys SP PLCWint V2.4.** Натиснути **OK** (рисунок 1).

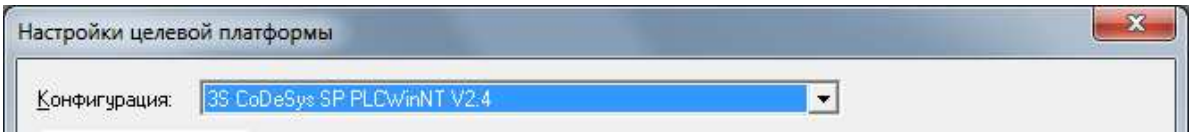

Рисунок 1 – Налаштування цільової платформи

Створення POU

У вікні діалогу, що з'явилося, визначите перший POU. За умовчанням він отримує найменування **PLC_PRG**. **Не змінюйте його**. Тип цього POU має бути -

ПРОГРАМА. Кожен проект повинен мати програму з таким ім'ям. В якості мови програмування цього POU виберемо мову Continuous Function Chart (рисунок 2).

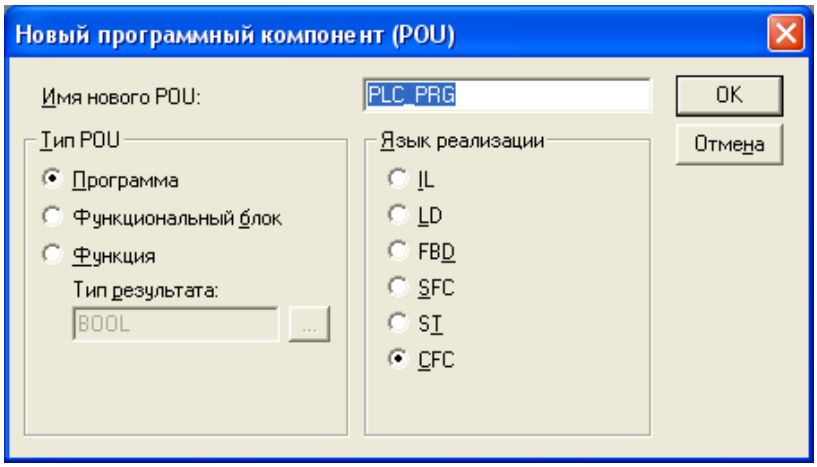

Рисунок 2 – Вікно діалогу створення POU PLC_PRG

У програмі PLC_PRG вводиться вхідний сигнал включення, що дозволяє початок роботи світлофора, і «колірні команди» кожної лампи пов'язані з відповідними виходами апаратури.

Створіть ще три POU. Скористайтеся командою **Проект - Об'єкт - Додати** в системному або в контекстному (натисніть праву кнопку миші (ПКМ) у вікні Організатора об'єктів - **Додати об'єкт.**) меню:

1) POU **Програма** на мові *Sequential Function Char t(***SFC**) з ім'ям **SEQUENCE** (рисунок 3).

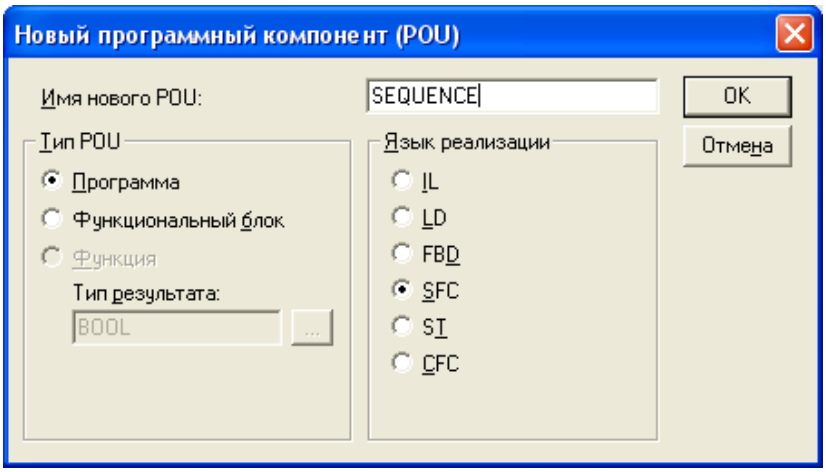

Рисунок 3 – Вікно діалогу створення POU SEQUENCE

У POU SEQUENCE все буде об'єднано так, щоб потрібні вогні запалювалися в правильний час і на потрібний період часу.

2) POU **Функціональний блок** на мові **FBD** з ім'ям **TRAFFICSIGNAL** (рисунок 4).

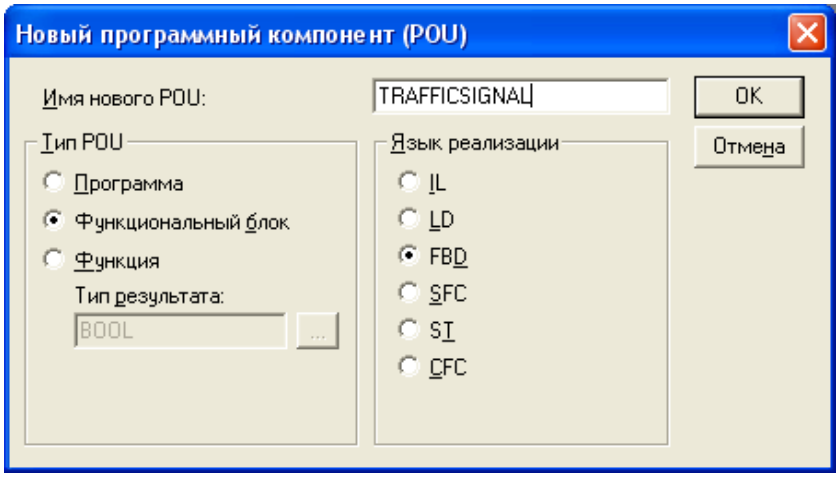

Рисунок 4 – Вікно діалогу створення POU TRAFFICSIGNAL

У POU TRAFFICSIGNAL зіставляються певні стадії процесу відповідним кольорам, т. е. червоне світло засвічене в червоній стадії і в жовто-червоній стадії, жовте світло в жовтій і жовто-червоній стадії і так далі.

3) POU **Функціональний блок** на мові **IL** з ім'ям **WAIT** (рисунок 5).

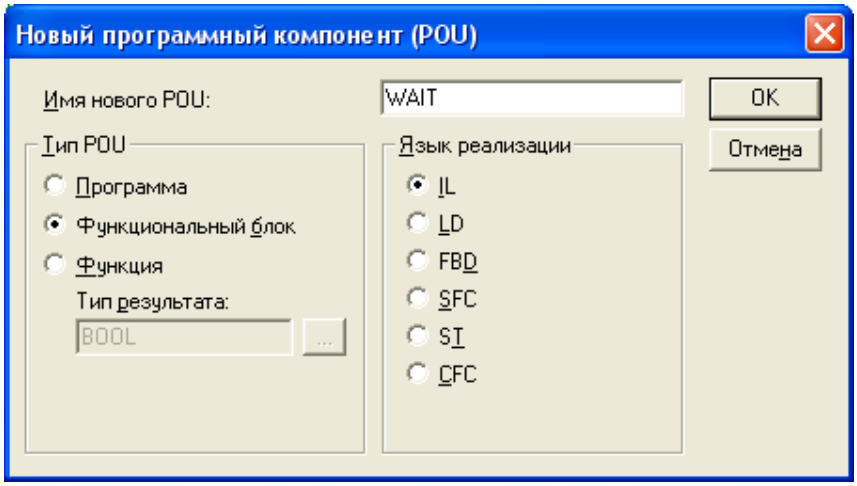

Рисунок 5 – Вікно діалогу створення POU WAIT

POU WAIT виконує функції таймера, який на вхід отримує довжину стадії в мілісекундах і на виході видає стан ІСТИНА після закінчення заданого періоду часу.

POU TRAFFICSIGNAL

Оголошення TRAFFICSIGNAL

Перейдіть до POU TRAFFICSIGNAL. У редакторові оголошень визначите вхідну змінну (між ключовими словами **VAR_INPUT** і **END_VAR**) на ім'я **STATUS** типу **INT**. Змінна STATUS матиме чотири можливі стани, що визначають відповідні стадії, - зелена (1), жовта (2), червона (3) і жовто-червона (4).

Блок TRAFFICSIGNAL повинен мати чотири виходи, три з яких управляють сигналами, а четвертий - станом світлофора. Для цього треба визначити ще чотири вихідних змінних (між ключовими словами **VAR_OUTPUT** і **END_VAR**) з іменами **RED**, **YELLOW**, **GREEN** і **OFF** типу **BOOL**. Тепер розділ оголошень TRAFFICSIGNAL- блоку повинен виглядати, як показано на рисунку.6.

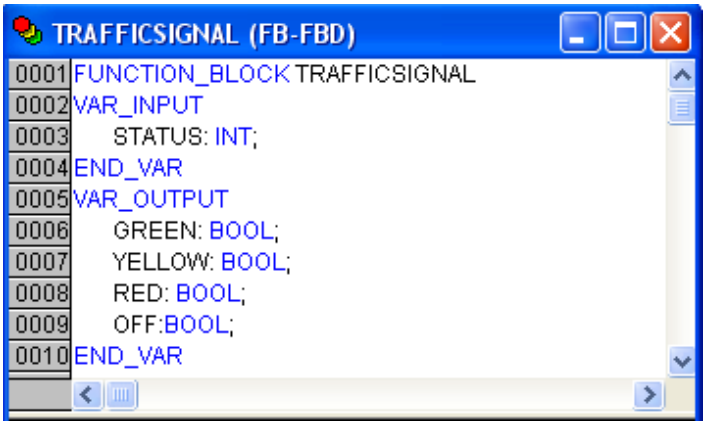

Рисунок 6 – Розділ оголошень функціонального блоку TRAFFICSIGNAL

Програмування TRAFFICSIGNAL

Необхідно зв'язати вхід STATUS з вихідними змінними. Для цього перейдіть в розділ програмного коду POU (тіло програми). Виділіть перший ланцюг, клацнувши на полі зліва від першого ланцюга (сіра область з номером 0001). На виділеному ланцюзі клацніть ПКМ і із спливаючого меню виберіть **Елемент** (чи команда меню **Вставити** – **Елемент** (Ctrl+B)).

У першому ланцюзі буде вставлений прямокутник з оператором AND і двома входами. Клацніть мишкою на тексті AND і замініть його на **EQ**. Три знаки питання біля верхнього з двох входів замініте на ім'я змінної **STATUS**. Для нижнього входу замість ??? треба поставити **1**. В результаті блок повинен виглядати так, як показано на рисунку 7.

Виділіть вихід блоку EQ(з правого боку блоку). Виконайте команду меню **Вставити - Привласнення(**Ctrl+A). Змініть три питання ??? на ім'я вихідної змінної **GREEN**. Отриманий ланцюг виглядатиме, як показано на рисунку 7.

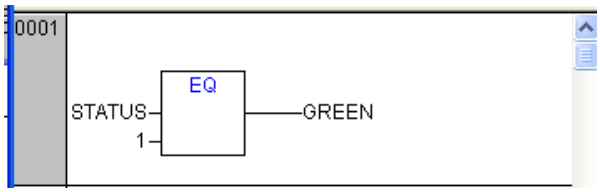

Рисунок 7 – Перший ланцюг

Цей ланцюг порівнюватиме значення змінної **STATUS** з **1**, а результат порівняння привласнюватиме змінній **GREEN**. Таким чином, GREEN буде включений, коли STATUS дорівнює 1.

Для інших вихідних змінних необхідно створити ще три ланцюги. Для їх створення виділите наявний ланцюг натисненням ПКМ і виберіть із спливаючого меню команду **Ланцюг (після).** Закінчений POU повинен виглядати, як показано на рисунку 8.

Примітка. Щоб вставити блок перед входом іншого блоку, треба виділити сам вхід (виділяється прямокутником), а не три питання. Далі використайте команду **Вставити - Елемент**.

POU WAIT

Підключення standard.lib

Для створення таймера в POU WAIT нам знадобиться POU із стандартної бібліотеки. Для цього відкрийте менеджер бібліотек командами **Вікно - Менеджер бібліотек**. Виберіть **Вставка - Додати бібліотеку**. Повинне відкритися діалогове вікно вибору файлів. Виберіть **STANDARD.lib** зі списку бібліотек.

Оголошення WAIT

Перейдіть до **POU WAIT.** Для блоку **WAIT** необхідно визначити наступні змінні:

1) Вхідна змінна **TIME_IN** типу **TIME** (змінні і константи типу TIME використовуються при роботі із стандартними модулями таймерів). Ця змінна зберігатиме значення тривалості стадії TRAFFICSIGNAL.

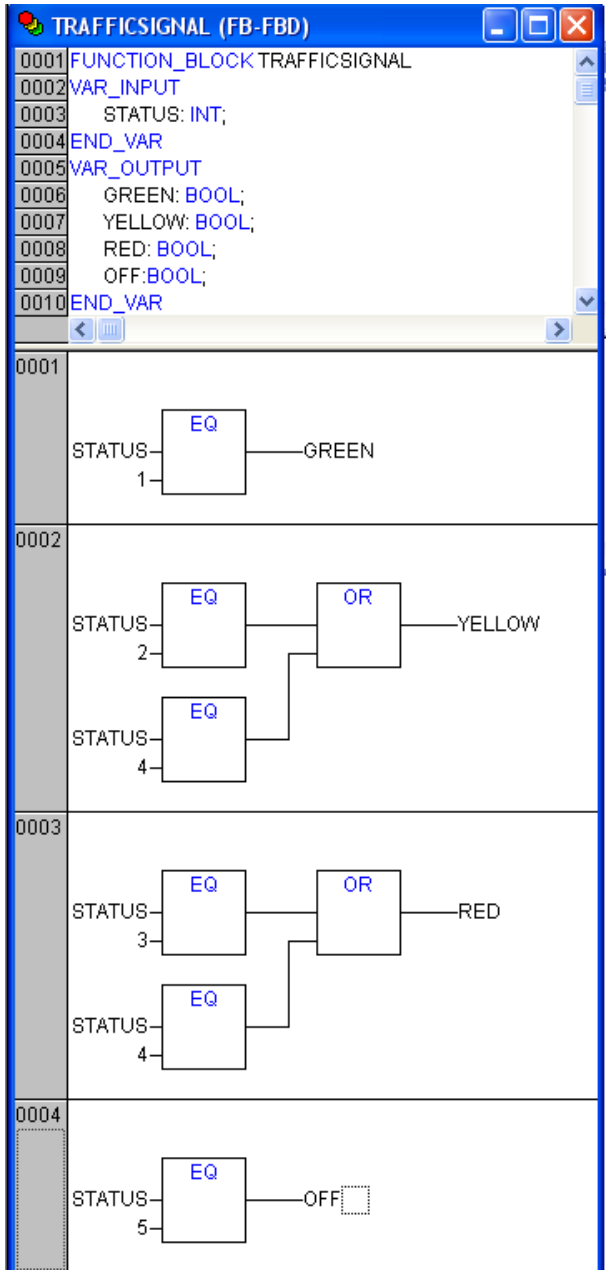

Рисунок 8 – Функціональний блок TRAFFICSIGNAL

2) Вихідна змінна **OK** типу **BOOL**. Ця змінна набуватиме значення **TRUE,** коли заданий в змінній **TIME_IN** період часу буде закінчений. Заздалегідь встановимо цю змінну в **FALSE** у кінці рядка оголошення (але до крапки з комою) ":= FALSE".

3) Локальна змінна(між ключовими словами **VAR** і **END_VAR**) **ZAB** (відлічений час) типу **TP**, яка є локальним екземпляром (копією) функціонального блоку "таймер **TP"** із стандартної бібліотеки середовища CoDeSys (опис цього таймера приведений в додатку 2).

Розділ оголошень WAIT тепер повинен виглядати як показано на рисунку 9.

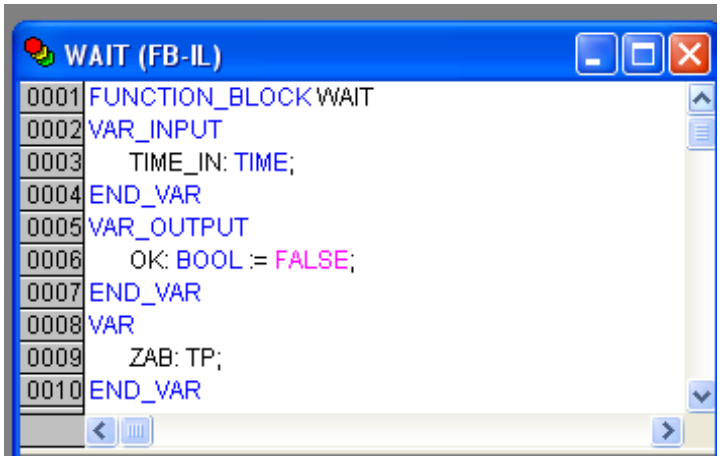

Рисунок 9 – Розділ оголошень WAIT

Програмування WAIT

Текст програми для створення таймера показаний на рисунку 10.

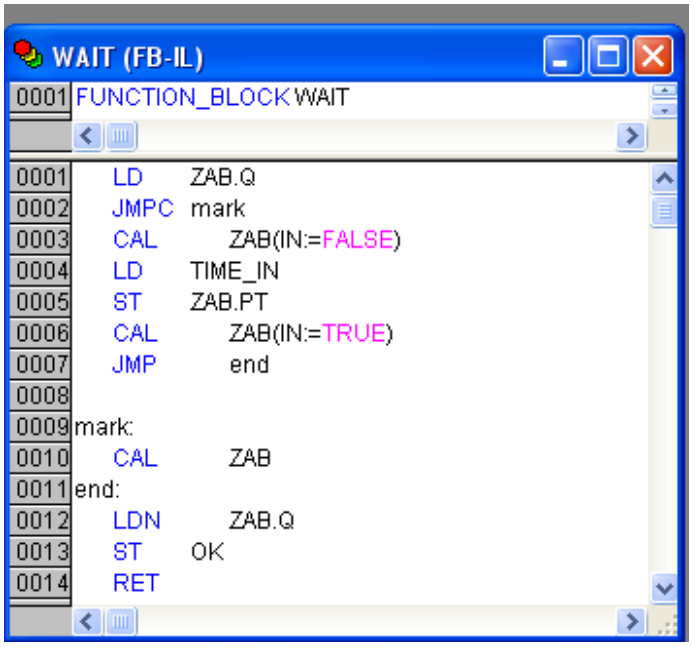

Рисунок 10 – Текст програми таймера

Програма працює таким чином. Спочатку у блоці перевіряється, чи встановлений вихід **Q** функціонального блоку **ZAB** в **TRUE**. Якщо ця умова виконується, то блок **ZAB** викликається без зміни значення входів для перевірки, чи закінчений період часу, заданий в змінній **TIME_IN** :

```
LD      ZAB.Q
JMPC    mark
```

Якщо ця перевірка не виконується, то вхід **IN** блоку **ZAB** встановлюється в **FALSE**, що дозволяє зупинити таймер:

```
CAL     ZAB(IN:=FALSE)
```

Встановлюємо значення часу, задане в змінній **TIME_IN**, яке повинен відпрацювати таймер блоку **ZAB**, на вході **PT** блоку **ZAB** :

```
LD      TIME_IN
ST      ZAB.PT
```

Після цього встановлюємо вхід **IN** блоку **ZAB** в стан **TRUE** і викликаємо **ZAB**:

```
CAL     ZAB(IN:=TRUE)
JMP     end
```

Інвертоване значення виходу **Q** блоку **ZAB** зберігатиметься в змінній **OK** після кожного виконання **WAIT**. Як тільки **Q** набуде значення **FALSE**, **OK** набуде значення **TRUE** :

```
    mark:
CAL     ZAB
    end:
LDN     ZAB.Q
ST      OK
RET
```

POU SEQUENCE

Оголошення SEQUENCE

Перейти на POU SEQUENCE. У нім спільно використовуватимуться раніше створені блоки TRAFFICSIGNAL і WAIT.

Необхідно визначити наступні змінні:

1) Вхідна змінна **START** типу **BOOL**.
2) Дві вихідні змінні **TRAFFICSIGNAL1** і **TRAFFICSIGNAL2** типу **INT.**
3) Локальна змінна: **DELAY** типу **WAIT**.

Програма SEQUENCE буде виглядати, як показано на рисунку 11.

Спочатку SFC діаграма завжди складається з кроку (етапу**) Init,** переходу **Trans0** і повернення назад до **Init**.

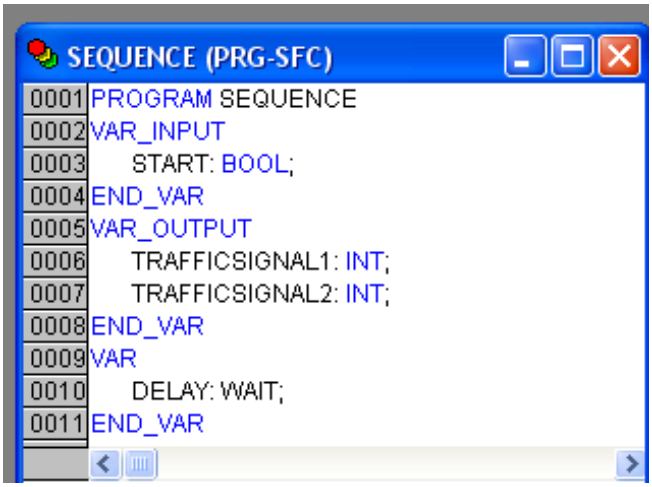

Рисунок 11 – Розділ оголошень програми SEQUENCE

Створіть кроки для кожної стадії TRAFFICSIGNAL. Вставте їх, відмічаючи Trans0 і вибираючи команди '**Вставити**' - '**Крок-перехід**'. Повторіть цю процедуру ще три рази.

Для редагування назви переходу або кроку треба просто клацнути мишкою на потрібному тексті. Назвіть перший перехід після Init **START**, а усі інші переходи **DELAY.OK**.

Перший перехід дозволяється, коли START встановлюється в TRUE, все ж інші - коли DELAY в OK стане TRUE, тобто коли заданий період закінчиться.

Кроки (зверху вниз) отримують імена **Switch1**, **Green2**, **Switch2**, **Green1**, Init, збереже своє ім'я. **Switch** повинен включати жовту фазу, в **Green1** TRAFFICSIGNAL1 буде зеленим, в **Green2** TRAFFICSIGNAL2 буде зеленим. Змініть адресу повернення Init на **Switch1**. В результаті діаграма повинна виглядати, як показано на рисунку 12.

Для програмування кроків і переходів використовуватимемо мову **IL (**Список Інструкцій).

Подвійне клацання мишею на зображенні кроку **Init** відкриває діалог визначення нової дії. Вибираємо IL і натискаємо OK (рисунок 13).

Під час дії кроку **Init** перевіряємо, активний сигнал включення START або ні. Якщо сигнал не активний, то світлофор вимикається. Цього можна досягти, якщо записати в змінні TRAFFICSIGNAL1 і TRAFFICSIGNAL2 число 5 (рисунок 14).

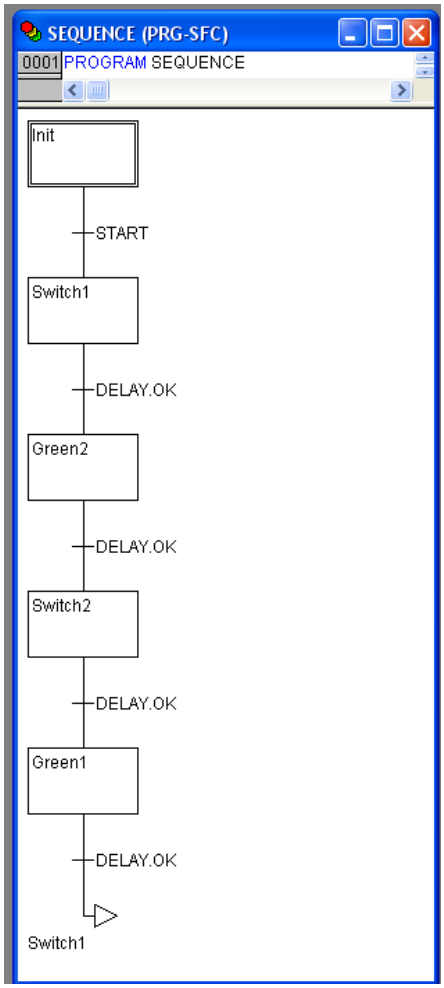

Рисунок 12 – Розділ інструкцій програми SEQUENCE

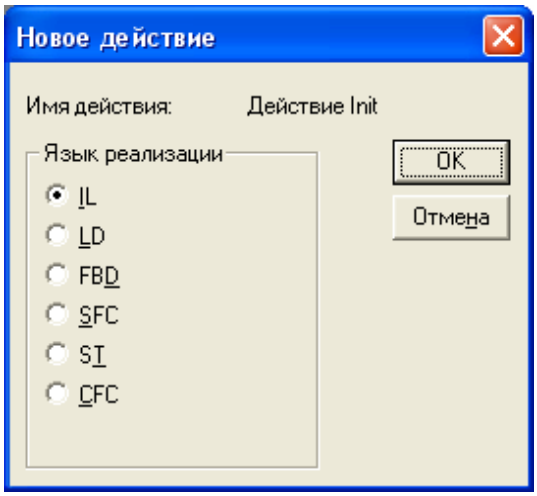

Рисунок 13 – Діалог визначення нової дії

Рисунок 14 – Дія кроку Init

Крок **Switch1** змінює стан TRAFFICSIGNAL1 на жовте (STATUS:=2) і, відповідно, TRAFFICSIGNAL2 - на жовто-червоне (STATUS:=4). Затримка перемикання сигналів складає 2 секунди (рисунок 15).

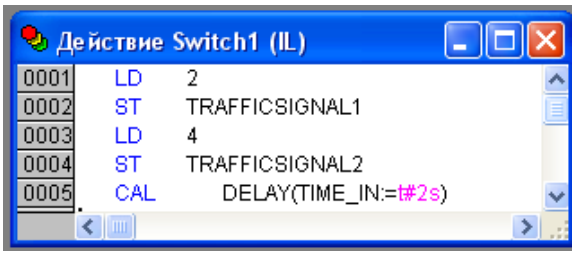

Рисунок 15 – Дія кроку Switch1

У **Green1** TRAFFICSIGNAL1 буде зеленим (STATUS:=1), TRAFFICSIGNAL2 буде червоним (STATUS:=3), затримка в 5 секунд (рис. 16).

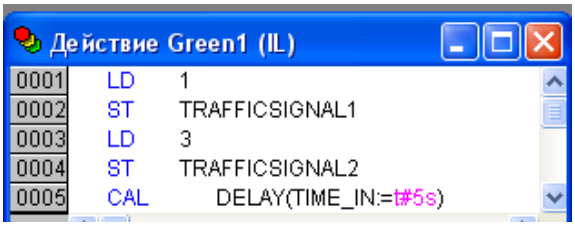

Рисунок 16 – Дія кроку Green1

У Switch2 STATUS в TRAFFICSIGNAL1 змінюється на 4 (жовто-червоний), відповідно, TRAFFICSIGNAL2 буде 2 (жовтий). Затримка тепер має бути в 2 секунди (рисунок 17).

Рисунок 17 – Дія кроку Switch2

Крок **Green2** включає червоний в TRAFFICSIGNAL1 (STATUS:=3) і зелений в TRAFFICSIGNAL2 (STATUS:=1). Затримка встановлюється в 5 секунд (рисунок 18).

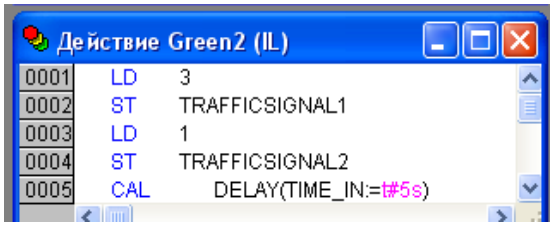

Рисунок 18 – Дія кроку Green2

На цьому етапі можна виконати перевірку роботи програми в режимі емуляції. Для цього відкрийте POU PLC_PRG. Кожен проект починає роботу з PLC_PRG. Вставте в нього компонент **Елемент** і замініте текст **AND** на **SEQUENCE**. Входи і виходи залиште доки вільними.

Відкомпілюйте програму командою **Проект - Компілювати** і перевірте відсутність помилок. У вікні повідомлень має бути текст: "**0 помилок, 0 попереджень".**

Виконайте команду **Онлайн - Підключення**. Запустіть програму **Онлайн - Старт**.

Відкрийте програму SEQUENCE. Програма запущена, але не працює, оскільки змінна START повинна мати значення TRUE. Далі це робитиме PLC_PRG, але зараз можна змінити її вручну. Для цього клацніть двічі мишачу по оголошенню цієї змінної. Її значення тепер виділене кольором і рівне TRUE. Дайте команду запису значень змінних (**Онлайн - Записати значення**). Тепер можна постежити за роботою програми. Активні кроки діаграми виділяються блакитним кольором.

Закрийте режим Онлайн командою **Онлайн - Відключення**.

Варіанти завдань

Варіанти завдань даної лабораторної роботи приведені в таблиці 2.

Таблиця 2 – Варіанти завдань

№ варіанту	Час роботи сигналів світлофора, сек	
	зелений/червоний	жовтий/червоно-жовтий
1	8	4
2	12	6
3	16	8
4	20	10
5	8	4
6	12	6

7	16	8
8	20	10
9	8	4
10	12	6
11	16	8
12	20	10
13	8	4
14	12	6
15	16	8
16	20	10
17	8	4
18	12	6
19	16	8
20	20	10

Контрольні питання

1. Мова IL.
2. Інструкція мови IL.
3. Акумулятор мови IL.
4. Модифікатори мови IL.
5. Виклик функцій IL.
6. Оператори LD та ST.
7. Арифметичні оператори IL.
8. Логічні оператори IL.
9. Оператори порівняння IL.
10. Оператор переходу на мітку.
11. Оператор виклику функції.
12. Оператор виходу із POU.

ЛАБОРАТОРНА РОБОТА № 7

Тема: МОДИФІКАЦІЯ СИСТЕМИ КЕРУВАННЯ СВІТЛОФОРОМ

Мета роботи: отримати практичні навички програмування на мові CFC, взаємодії POU, налагоджування програмного забезпечення.

Постановка задачі: виконати модифікацію розробленої системи керування світлофором.

Короткі теоретичні відомості

Мова програмування CFC

Мова **CFC (**Continuous Function Chart - безперервні функціональні схеми) - це графічна мова програмування, програма, точніше POU, який нагадує блок-схему алгоритму.

Основні компоненти, з яких створюється POU :

Елемент Вхід Вихід Перехід Мітка Повернення Коментар

Вставка усіх компонентів здійснюється у вікні операторів POU таким чином: ПКМ - вибрати компонент зі списку - перетягнути компонент на потрібне місце - ЛКМ.

Кожен вставлений компонент можна редагувати, міняючи його ім'я, тип, кількість входів, виходів і так далі

Порядок виконання CFC-схеми

Кожен компонент схеми має номер, який вказує порядок його виконання (!!!).

При створенні або вставці елементу він автоматично отримує номер відповідно до наступного правила: зліва направо **і зверху вниз.** Номер елементу не змінюється при його переміщенні.

Послідовність дій визначає результат і має бути змінена при необхідності.

Номер відображається в правому верхньому кутку елементу, якщо включений режим відображення.

Для зміни порядку елементів слід поступити таким чином: ПКМ в розділі операторів POU - Порядок - вибрати необхідну команду зі списку:

Показати порядок

Упорядкувати топологічно

Відповідно до потоку даних

Порядок: вище

Порядок: нижче

Порядок: в початок

Порядок: в кінець

Основні компоненти мови CFC

Елемент  – використовується для вставки операторів, функцій, функціональних блоків і програм;

– за умовчанням вставляється екземпляр функціонального блоку **AND**;

– у текстовому полі блоку можна вказати ім'я будь-якого іншого оператора, функції, функціонального блоку або програми;

– графічний редактор автоматично промальовував необхідну кількість входів/виходів нового блоку і означає їх.

Примітка. Для вибору імені змінною, функції, функціонального блоку або програми слід натиснути **F2**.

Вхід  - **??**? слід замінити ім'ям вхідної змінної або константи.

Вихід  - **???** слід замінити ім'ям змінної, якою буде присвоєно вихідне значення.

Повернення  - вставка команди **Return**.

Порядок виконання роботи

Постановка задачі. Необхідно передбачити в розробленої програмі виключення світлофорів на ніч. Для цього в програмі створюється лічильник, який після деякого числа циклів блоку TRAFFICSIGNAL проведе відключення пристрою.

Рішення задачі

Відкрити проект, розроблений на попередній лабораторній роботі.

Редагування POU SEQUENCE

Оголосіть в розділі оголошень програми SEQUENCE нову змінну **COUNTER** (лічильник) типу **INT,** яка зберігатиме значення кількості циклів SEQUENCE, що повторилися.

Відредагуйте структуру SFC діаграми SEQUENCE, виконавши наступні дії:

1) додайте новий крок і перехід, виділивши перехід **DELAY.OK** після кроку **Switch1** натисненням ПКМ і з меню, що з'явилося, вибравши команду **Крок-перехід знизу**;

2) додайте альтернативну гілку, виділивши перехід **Trans0** натисненням ПКМ і з меню, що з'явилося, вибравши команду **Альтернативна гілка справа**;

3) додайте новий крок і перехід в лівій гілці, виділивши перехід **Trans0** натисненням ПКМ і з меню, що з'явилося, вибравши команду **Крок-перехід знизу**;

4) додайте безумовний перехід, виділивши перехід **Trans3** натисненням ПКМ і з меню, що з'явилося, вибравши команду **Безумовний перехід**;

5) видаліть перехід **Trans2** і крок **Step1**;

6) замініть імена кроків **Step0, Step2** на відповідні імена **COUNT** і **OFF**, переходи **Trans0**, **Trans1** і **Trans3** на **EXIT**, **TRUE**, **DELAY.OK** відповідно, безумовний перехід **Step** на **SWITCH1**.

Тепер нові частини повинні виглядати як фрагмент, показаний на рисунку 1.

Програмування

На кроці **Count** виконується тільки одна дія - **COUNTER** збільшується на **1** (рисунок 2).

На переході **EXIT** перевіряється досягнення лічильником заданого значення, наприклад **7** (рисунок 3).

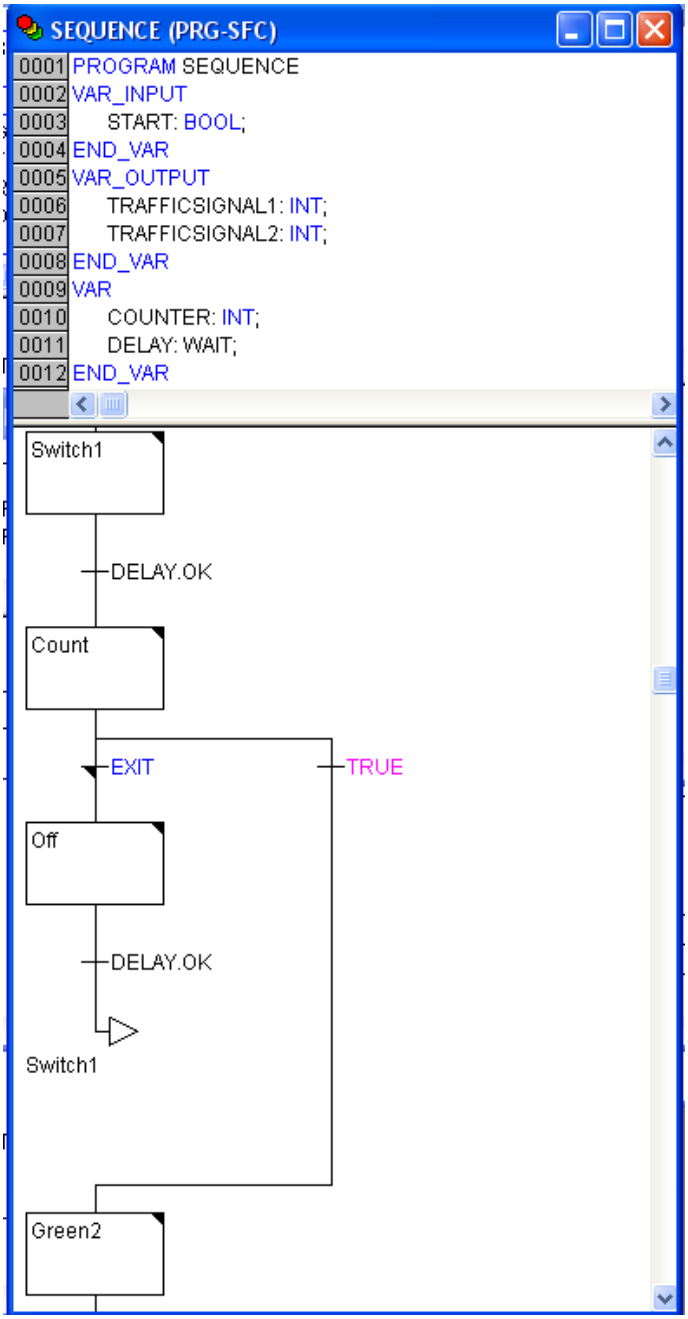

Рисунок 1 – Фрагмент програми SEQUENCE після змін

Рисунок 2 – Дія Count

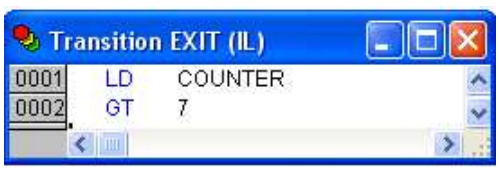

Рисунок 3 – Перехід EXIT

На кроці **Off** стан обох світлофорів встановлюється в 5 (світлофор вимкнений), **COUNTER** скидається в 0 і встановлюється затримка часу в 10 секунд (рисунок 4).

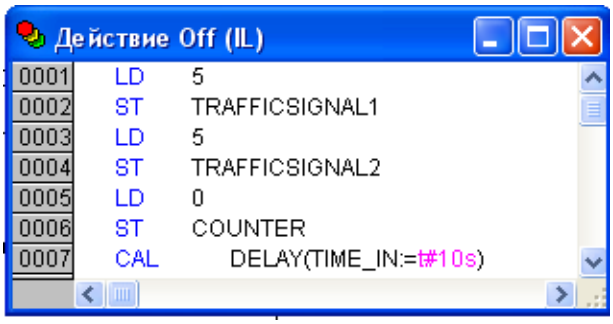

Рисунок 4 – Дія Off

Результат

У цій гіпотетичній ситуації ніч настає після семи циклів TRAFFICSIGNAL. Світлофори повністю вимикаються до світанку, і процес повторюється знову. На цьому етапі ще раз перевірте роботу програми в емуляторі.

Основна програма PLC_PRG

В основному POU PLC_PRG необхідно створити програму, в якій буде реалізований запуск системи управління вимикачем **IN** і забезпечено перемикання усіх шести ламп (2 світлофори) шляхом передачі «команд перемикання» на кожному кроці SEQUENCE.

Необхідно оголосити відповідні логічні змінні для усіх шести виходів (шість ламп світлофорів) і одного входу (вимикач **IN**), потім створимо програму і зіставимо змінні відповідним IEC адресам.

Оголошення PLC_PRG

Необхідно оголосити змінні **LIGHT1** і **LIGHT2** типу **TRAFFICSIGNAL** в редакторові оголошень (рисунок 5).

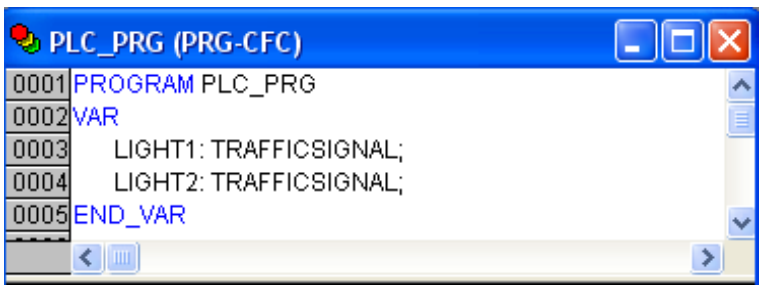

Рисунок 5 – Оголошення змінних LIGHT1 і LIGHT2

Оголосимо глобальні змінні:

1) шість змінних типу **BOOL** для представлення шести ламп світлофорів;
2) двійкову вхідну змінну **IN**, необхідну для установки змінної START блоку SEQUENCE в TRUE.

Для цього виберіть вкладку **Ресурси** Організатора об'єктів і відкрийте список **Глобальні змінні**. Подвійне клацання по **Global_Variables** відкриє вікно оголошення глобальних змінних (рисунок 6).

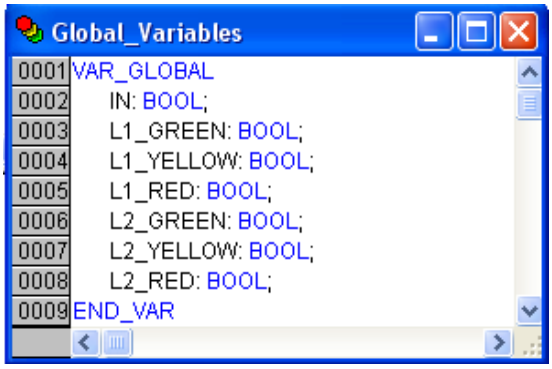

Рисунок 6 – Оголошення глобальних змінних

Програмування PLC_PRG на мові CFC

Для цього перейдіть у вікно редактора **CFC** і буде доступна відповідна панель інструментів.

Клацніть ПКМ у вікні редактора і виберіть **Елемент**. Клацніть на тексті **AND** і замініть на **SEQUENCE**. Елемент автоматично перетвориться в SEQUENCE із вже певними вхідними і вихідними змінними.

Вставте далі два елементи і назвіть їх **TRAFFICSIGNAL**. TRAFFICSIGNAL - це функціональний блок, у якого три червоні знаки питання треба замінити вже оголошеними локальними змінними **LIGHT1** і **LIGHT2**.

Створіть компонент типу **Вхід,** який дістане назву **IN** і шість компонентів типу **Вихід,** яким треба дати наступні імена : **L1_green**, **L1_yellow**, **L1_red**, **L2_green**, **L2_yellow**, **L2_red**.

З'єднайте входи і виходи елементів програми. Для цього клацніть мишею на короткій лінії входу/виходу і тягніть її (не відпускаючи клавішу миші) до входу/виходу потрібного елементу.

Програма повинна набрати вигляду, показаного на рисунку 7.

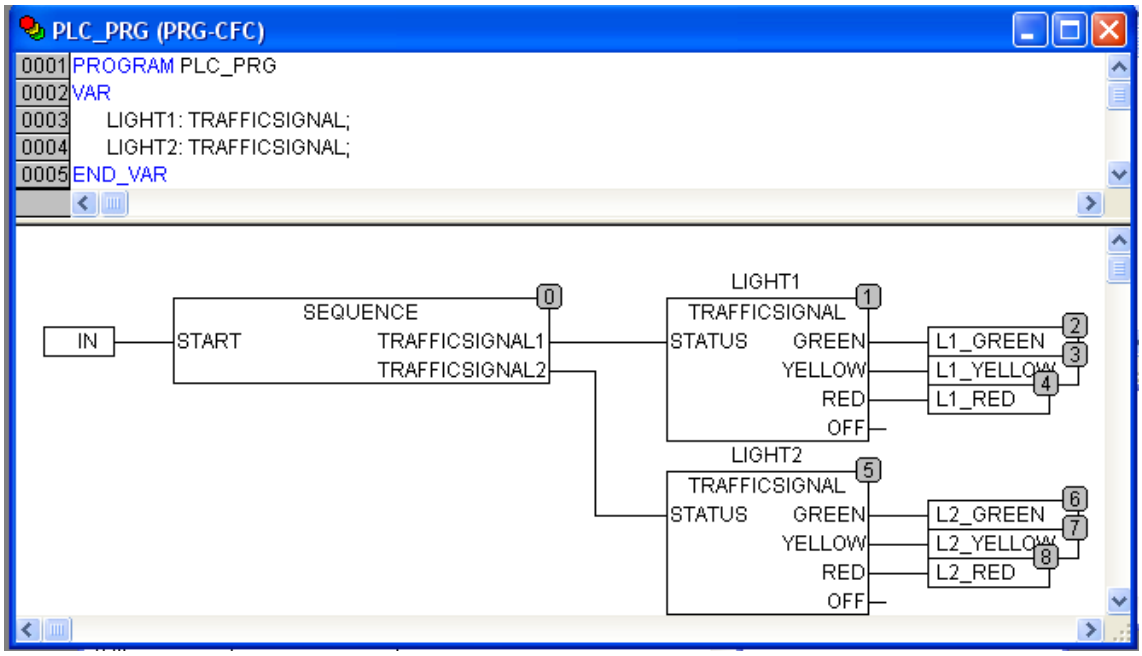

Рисунок 7 – Програма PLC_PRG

Емуляція TRAFFICSIGNAL

Перевірте остаточно програму в режимі емуляції. Відкомпілюйте проект (**Проект - Компілювати**) і завантажте його (**Онлайн - Підключення**). Запустіть програму командою **Онлайн - Старт**. Потім встановите змінну **IN** в **TRUE**, наприклад подвійним клацанням мишки по входу **IN** в CFC редакторові. Переконайтеся, що праворуч від змінної з'явилося нове значення <**TRUE**>, підготовлене для запису в контроллер. Для запису значення натисніть <Ctrl><F7> чи дайте команду **Онлайн - Записати значення**. Тепер змінна **START** в **SEQUENCE** (яку ми встановлювали вручну в TRUE при виконанні завдання 1) набуде значення **IN** з **PLC_PRG**. Це приведе до запуску циклу роботи світлофора. Робота PLC_PRG відображається тепер у вікні моніторингу. Клацніть двічі по значку плюс у вікні оголошень. Список змінних буде розкритий і можна побачити значення окремих змінних.

Контрольні питання

1. Мова CFC. CFC-схема.
2. Порядок виконання CFC-схеми.
3. Як змінити порядок CFC-схеми?
4. Які основні компоненти мови CFC?

ЛАБОРАТОРНА РОБОТА № 8

Тема: РОЗРОБКА ІНТЕРФЕЙСУ ОПЕРАТОРА ДЛЯ СИСТЕМИ КЕРУВАННЯ СВІТЛОФОРОМ

Мета роботи: Практичне закріплення знань про програмування на мовах FBD, IL, SFC та CFC, взаємодії POU, налагоджування програми, побудови інтерфейсу оператора (візуалізації).

Постановка задачі: Виконати візуалізацію проєкту системи управління світлофорами.

Порядок виконання роботи

Постановка задачі. За допомогою візуалізації необхідно намалювати два світлофори і їх вимикач, який дозволить включати і вимикати блок управління світлофором.

Рішення задачі

Створення нової візуалізації

Відкрити проект, виконаний на лабораторній роботі № 7.

Для того, щоб створити візуалізацію, виберіть вкладку **Візуалізація** в Організаторові об'єктів. Натисніть ПМК і виберіть команду **Додати об'єкт**. У вікні, що з'явилося, введіть будь-яке ім'я для візуалізації, наприклад *Lights* (рисунок 1).

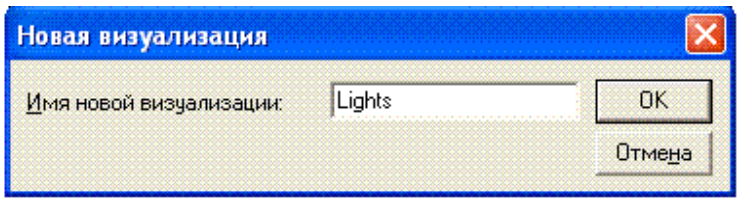

Рисунок 1 – Діалог для створення нової візуалізації

Натисніть кнопку **ОК**, відкриється вікно, в якому створюватимете візуалізацію.

Вставка графічних елементів у візуалізацію

Перший світлофор

Для створення візуалізації світлофора виконаєте наступні дії:

- Виберіть команду **Вставка - Еліпс** і намалюйте коло з діаметром близько 2 сантиметрів. Для цього клацніть мишею на робочому полі і, утримуючи ліву кнопку миші, розтягніть коло, що з'явилося, до необхідного розміру.
- Двічі клацніть мишею на колі. З'явиться діалогове вікно для налаштування елементу візуалізації.
- Виберіть категорію **Змінні** і в полі **Зм. кольору** введіть ім'я змінної **.L1_red**. Вводити ім'я змінної зручно за допомогою *Асистента введення (*клавіша <F2>). Глобальна змінна **L1_red** управлятиме кольором намальованого кола (рисунок 2).

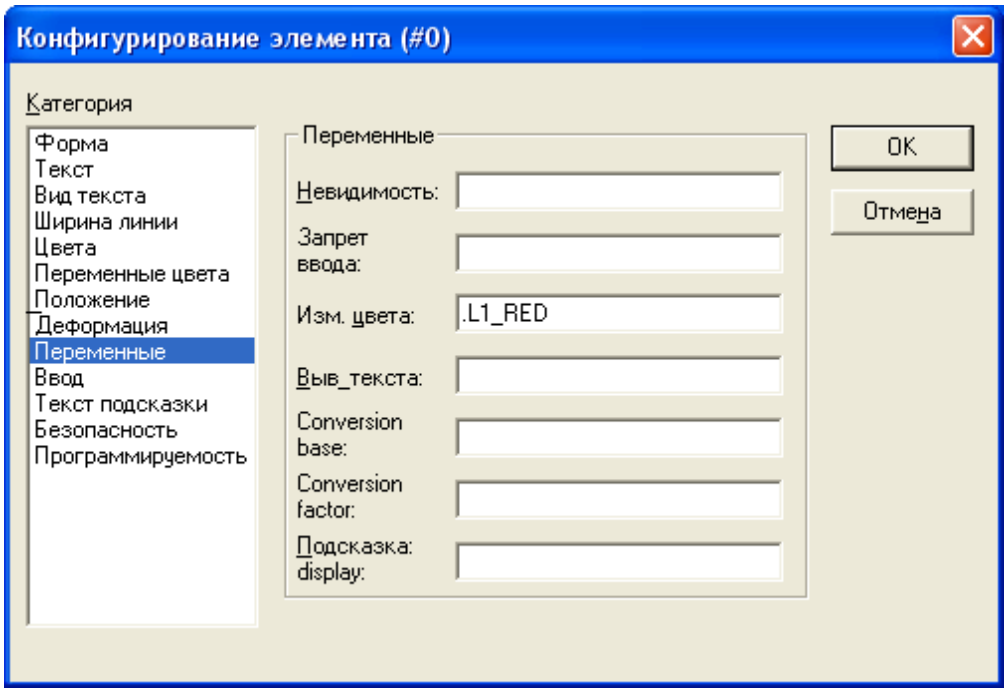

Рисунок 2 – Налаштування елементу візуалізації

- Виберіть категорію **Кольори**. У області **Кольори** натисніть кнопку **Заливка** і у вікні, що з'явилося, виберіть будь-який нейтральний колір, наприклад, чорний.
- Натисніть кнопку **Заливка** в області **Тривожний колір** (рисунок 3) і виберіть червоний колір.

Отримане коло буде чорним, коли значення змінної неправдиве, і червоною, коли змінна істинна.

Таким чином, ми створили перший ліхтар першого світлофора.

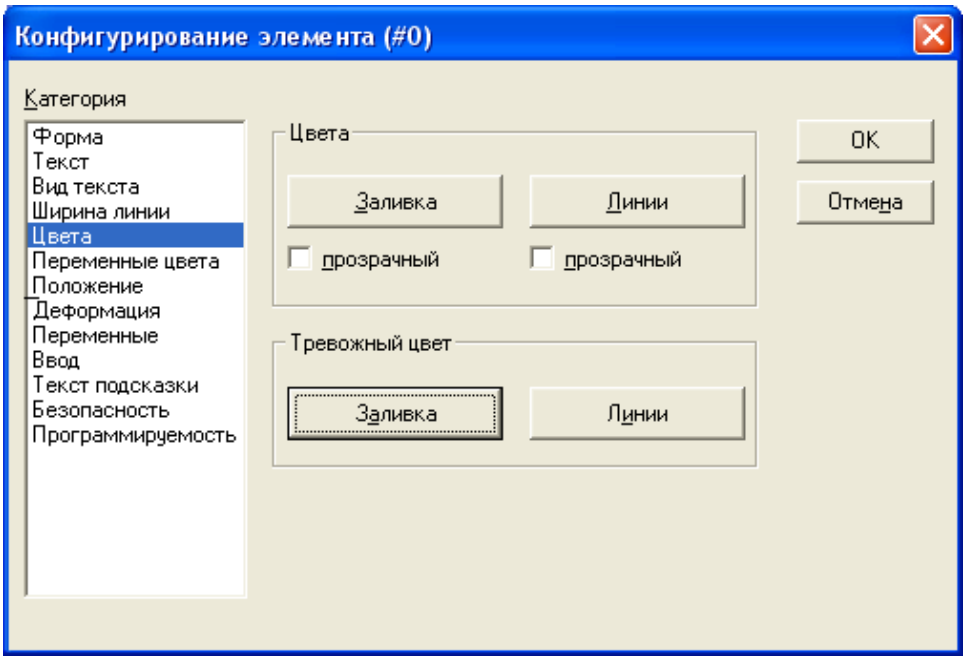

Рисунок 3 – Налаштування кольору елементу

Для створення інших кольорів світлофора викличте команду копіювання **Правка – Копіювати** (<Ctrl>+<C>) і двічі виконаєте команду вставки **Правка – Вставити** (<Ctrl>+<V>). Отримайте два нові кола. Переміщати ці кола можна за допомогою мишки. Розташуйте їх так, щоб вони були вертикальним рядом в лівій частині вікна редактора.

Подвійне клацання по колу призводить до відкриття вікна для налаштування властивостей елементу візуалізації. У полі **Зм. кольору** діалогу **Змінні** вікон налаштування властивостей відповідних кіл введіть наступні змінні:

для середнього кола: **.L1_yellow**

для нижнього кола: **.L1_green**

У категорії **Кольори** в області **Тривожний колір** встановите кольори кіл (жовтий і зелений).

Корпус світлофора

Викличте команду **Вставка - Прямокутник** і вставте прямокутник так, щоб введені раніше кола знаходилися усередині нього. Виберіть колір прямокутника і потім виконайте команду **Доповнення - На задній план**, яка перемістить його на задній план. Після цього кола знову будуть видні.

Запустіть програму в режимі емуляції шляхом виконання команд **Онлайн - Підключення** і **Онлайн - Старт**, і спостерігайте, як мінятимуться кольори світлофора.

Другий світлофор

Для створення другого світлофора треба скопіювати усі елементи першого. Виділіть елементи першого світлофора і скопіюйте їх, виконавши команди **Правка - Копіювати** і **Правка - Вставити**. Замініть імена змінних, що управляють кольорами(наприклад, **.L1_red** на **.L2_red**), і другий світлофор буде готовий.

Перемикач ON

Як описано вище, вставте прямокутник для перемикача **ON.** У полі **Рядок** категорії **Текст** введіть ім'я **ON** (рисунок 4).

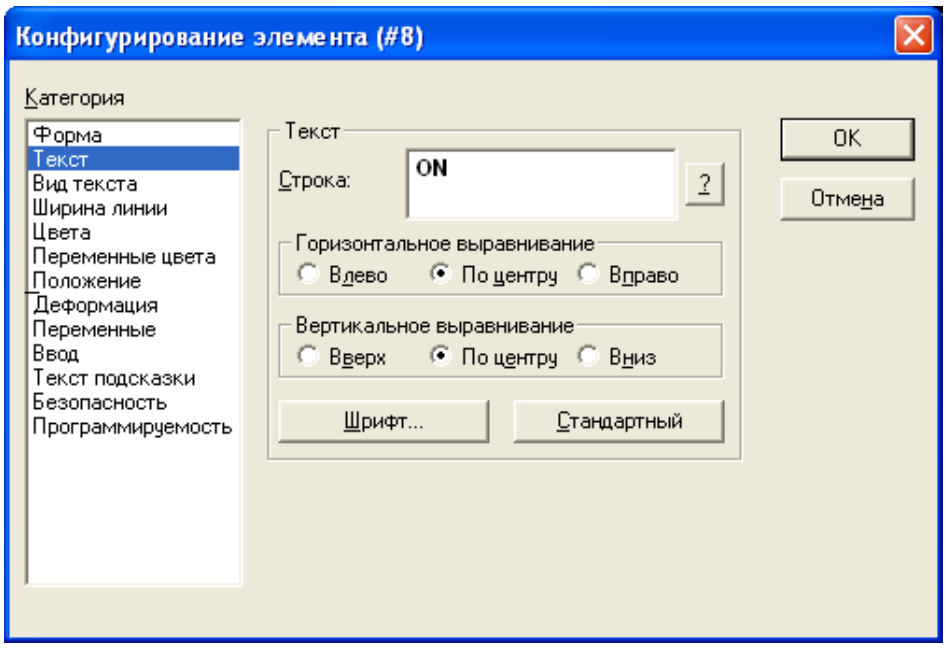

Рисунок 4 – Заповнення категорії Текст для перемикача

Встановіть його колір в категорії **Кольори**.

Введіть змінну **.IN** в полі **Зм. кольору** категорії **Змінні** (рисунок 5).

Для того, щоб змінна ON перемикалася при клацанні мишкою на цьому елементі, в полі **Зм-а перемикання** категорії **Введення** введіть змінну **.IN**. Створений перемикач включатиме/вимикати світлофори (рисунок 6).

Відобразити включений стан кнопки можна кольором, як і для світлофора. Виберіть категорію **Кольори,** натисніть кнопку **Заливка** в області **Тривожний колір** і виберіть будь-який колір, наприклад, ясно-зелений.

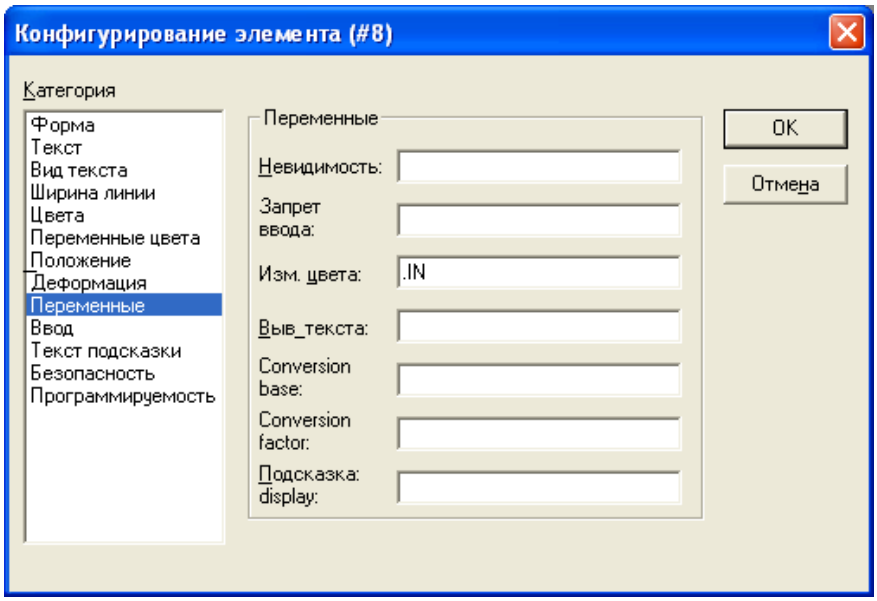

Рисунок 5 – Заповнення категорії Змінні для перемикача

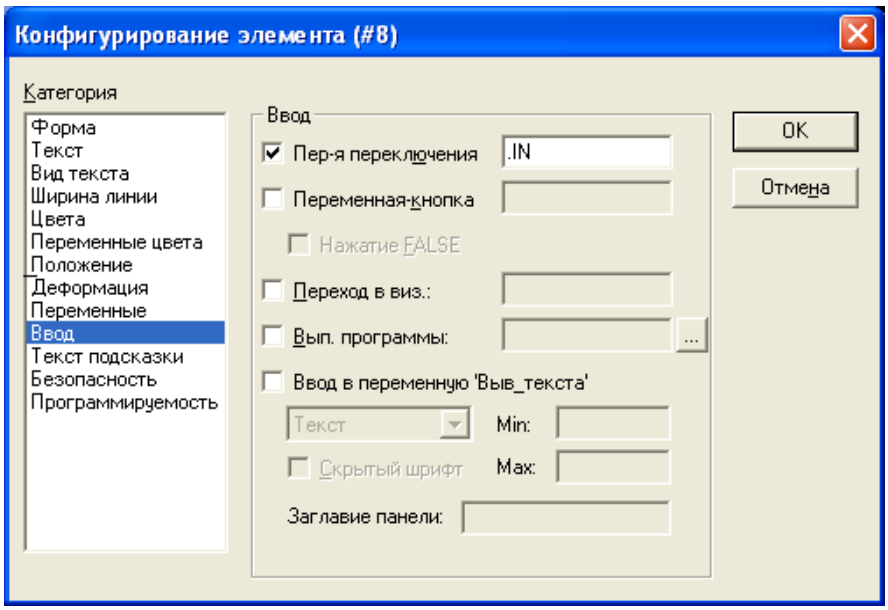

Рисунок 6 – Налаштування перемикання

Написи у візуалізації

Під світлофорами вставимо два прямокутники.

У властивостях елементу в категорії **Кольори** колір **Лінії** прямокутника задайте білим. У полі **Рядок** (категорія **Текст**) введіть назви світлофорів **Light1** і **Light2** (рисунок 7).

Перевірка роботи світлофора

Виконайте завантаження і запуск програми. Викличте вікно **Lights**, натисніть кнопку **ON**. При цьому повинен запуститься світлофор (рисунок 8).

Поспостерігайте виконання усіх програм в режимі Онлайн. Відкрийте **Дію Off (IL)**, простежте за лічильником **COUNTER**, як він вимкне світлофори після семи циклів і через 10 з включите знову.

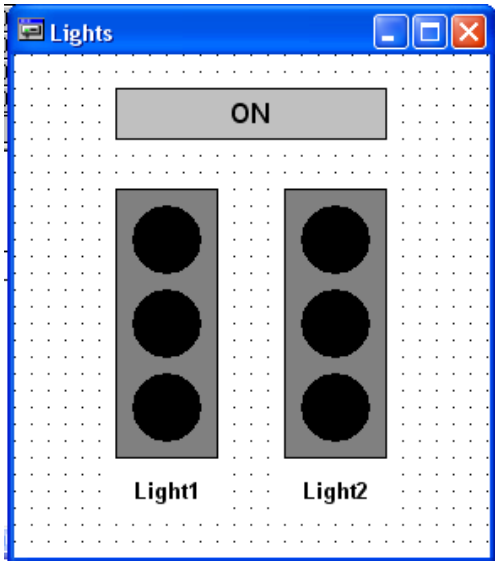

Рисунок 7 – Візуалізація проєкту

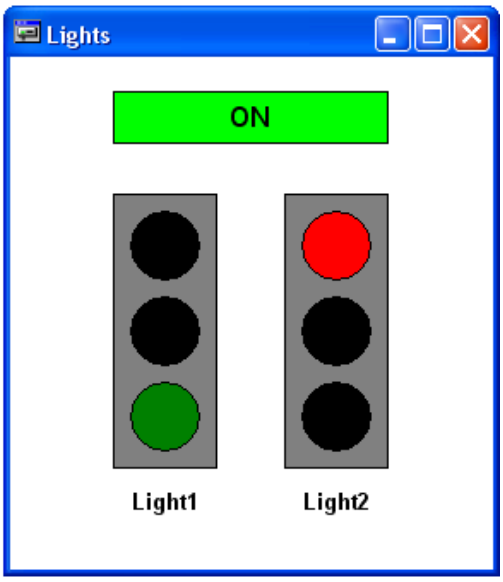

Рисунок 8 – Перевірка роботи світлофора

Контрольні питання

1. Пояснить принцип роботи створеної системи керування.
2. Які графічні елементи візуалізації були використані?
3. Як виконувалось конфігурування графічних елементів?

ЛАБОРАТОРНА РОБОТА № 9

Тема: РОЗРОБКА СИСТЕМИ КОНТРОЛЮ ТЕМПЕРАТУРИ

Мета роботи: вивчення функцій програмованого логічного контролера ПЛК150, можливостей використання його в системах контролю і управління технологічними процесами, а також отримання практичних навичок установки таргет-файлів в системі CoDeSys і додавання їх до проєкту.

Постановка задачі: розробити віртуальну установку контролю температури.

Короткі теоретичні відомості

Призначення і функції програмованого логічного контролера ПЛК-150

Програмований логічний контролер ОВЕН ПЛК150 виробництва фірми ОВЕН призначений для створення систем автоматизованого управління технологічним устаткуванням в різних галузях промисловості. Зовнішній вигляд лицьової панелі контролера із запущеною програмою може мати вигляд, показаний на рисунку 1.

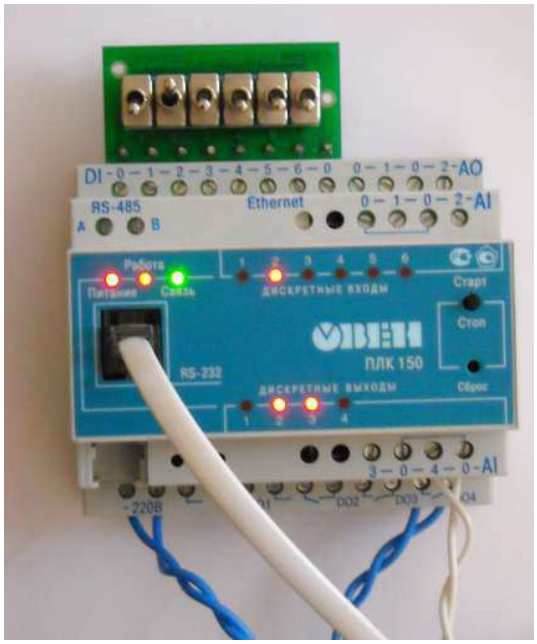

Рисунок 1 – Зовнішній вигляд лицьової панелі контролера із запущеною програмою

Програмують контролер за допомогою інтегрованого середовища CoDeSys компанії 3S_Smart Software Solutions. Після інсталяції середовища CoDeSys слід виконати інсталяцію Target-файлів. У Target-файлах міститься інформація про ресурси програмованих контролерів, з якими працює CoDeSys. Target-файл поставляється виробником контролера

Не завжди користувачеві потрібні усі таргет-файли. Реально використовуються зазвичай 2-3 модифікації. Можна вибрати тільки ті модифікації, які вам дійсно потрібні. Для цього необхідно відкрити вміст папки TARGET-files, ви виявите вкладені папки з англомовними найменуваннями ПЛК. Приміром, розглянута вище модифікація ПЛК110-30.Р-L, тут матиме назву PLC110.30_l (рисунок 2).

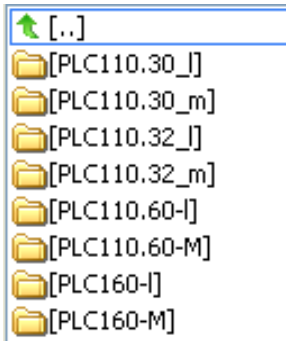

Рисунок 2 – Вміст папки TARGET-files

Наприклад, для *версії «прошивки»* 2.11 (тобто внутрішнє програмне забезпечення ПЛК) :

ПЛК150-220.I-L = target PLC150 - I - L

ПЛК150-220.У-L = target PLC150 - U - L

Потрібні target-файли можна скачати самостійно. Ця процедура здійснюється за допомогою програмного модуля *InstallTarget.exe*, який знаходиться в тій директорії, що і файл запуску середовища CoDeSys 2.3 (рисунок 3).

Рисунок 3 – Шлях до модуля *InstallTarget.exe*

Далі необхідно діяти таким чином. За допомогою кнопки «…». у верхній частині вікна вибрати потрібну директорію з target-файлами для установки c:\02. target-файлы і натиснути на кнопку ОК. Використовуючи кнопку Open ..., знайти каталог 02. PLC110_160 з потрібним target- файлом. Послідовно розкрити теки Ver. 2.10, Manual Installation і PLC110.30_l, в останній - відкрити файл plc.tnf (рисунок 4).

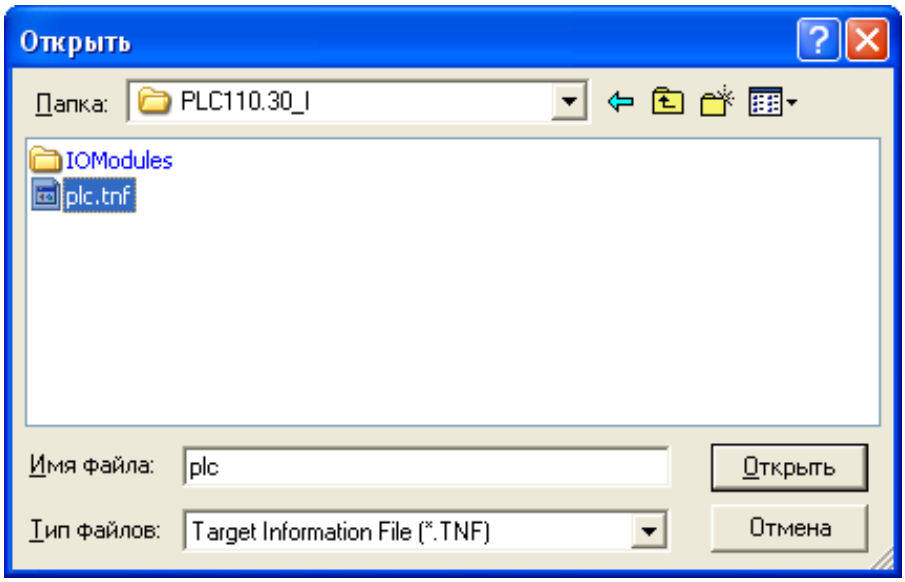

Рисунок 4 – Вибір файлу plc.tnf

В результаті в лівій частині вікна Possible Targets з'явиться директорія Owen. Розкрити директорію Owen, виділити файл PLC110.30 - I, натиснути на кнопку Install і далі, переконавшись, що потрібний файл з'явився в правій частині вікна в теці Owen - натиснути на кнопку Close. На рисунку 5 зображено вікно програми-установника InstallTarget з потрібним target-файлом перед його завантаженням.

Якщо вибрати вже встановлену цільову платформу ПЛК в списку справа, потім натиснути кнопку «Remove», платформа віддалиться зі списку встановлених, і після цього для неї не можна буде створювати проекти. Для відновлення такої можливості треба буде цю платформу повторно встановлювати.

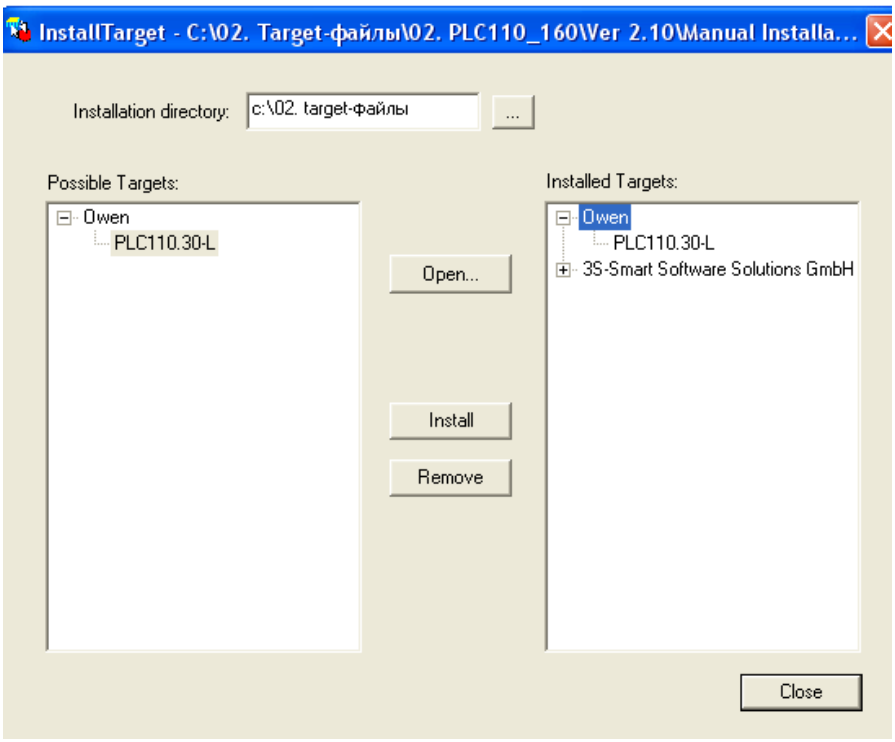

Рисунок 5 – Вікно з потрібним target- файлом перед установкою

Порядок виконання роботи

Постановка задачі. Розробити віртуальну установку, у рамках якої необхідно контролювати температуру і необхідно відстежувати, щоб вона не перевищувала заданого значення певної уставки, наприклад 80°С. У разі, якщо температура піднімається вище заданої уставки і тримається впродовж 3 с, то спрацьовує сигналізація - лампа, яка починає блимати і, відповідно, спрацьовує деяка охолоджувальна установка. Коли температура падає, сигналізація охолоджувальної установки продовжує працювати, до тих пір, поки їх роботу не скинемо кнопкою «Сброс». При натисненні на кнопку «Сброс» все відключається і система продовжує працювати далі до наступного перевищення температури. Уставку в системі можна міняти.

Рішення задачі

Створення проєкту

Створити новий проект в системі CoDeSys. Для цього виберіть **'Файл' - 'Створити'.**

Вибрати цільову платформу «PLC 150.I-L» (рисунок 6) і підтвердити вибір кнопкою «ОК».

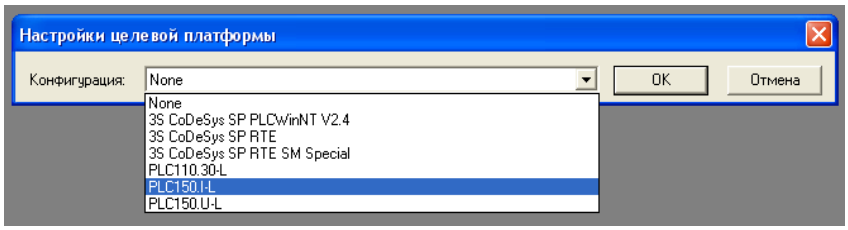

Рисунок 6 – Вибір цільової платформи ПЛК

На екрані з'явиться нове вікно, в якому міститимуться основні параметри і налаштування вибраної платформи ПЛК (адреси сегментів пам'яті, тактова частота процесора, тип процесора, кількості входів і виходів, значення деяких системний змінних), деякі з яких користувач може змінити. У нашому випадку міняти нічого не потрібно, треба тільки натиснути кнопку «ОК» (рисунок 7).

У вікні створення нового POU, що з'явилося, його тип Програма і ім'я PLC_PRG залишити без зміни, вибрати мову реалізації CFC, натиснути «ОК».

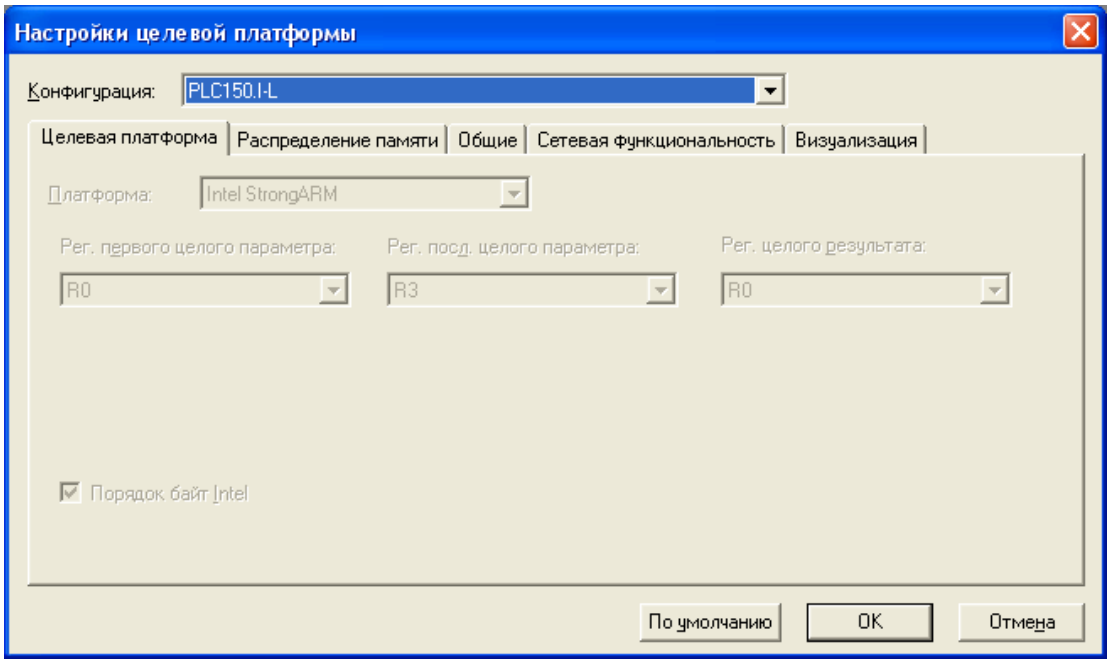

Рисунок 7 – Налаштування цільової платформи ПЛК

Конфігурування ПЛК

Перейдіть на вкладку **Ресурси**, виберіть утиліту **Конфігурація ПЛК**. На вкладку **Параметри модуля** задають час циклу - 10 мс. Далі клік миші по знаку

«+», який знаходиться зліва поруч зі слотом *PLC150.I-L*, відкрийте його вміст. Появиться дерево з фіксованим набором ресурсів ПЛК (рисунок 8).

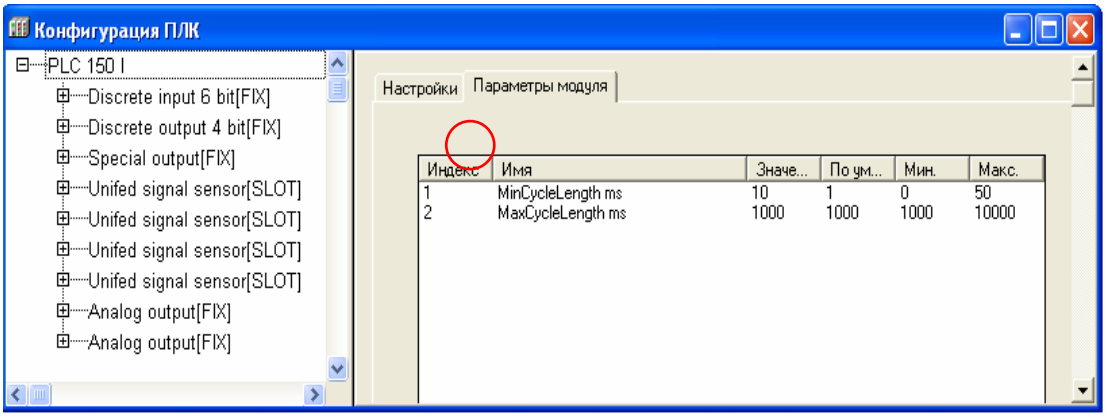

Рисунок 8 – Завдання параметрів модуля

Зберегти проект.

Виконати конфігурація каналів дискретних входів / виходів і аналогових входів контролера. Для вирішення поставленого завдання знадобитися 1 дискретний вхід - кнопка скидання (змінна **sbr**), 2 дискретних виходу - лампа сигналізації, яка будуть мигати, і запуск системи охолодження (змінні **lamp** і **ohl**). Для цього необхідно навпаки відповідного рядка в конфігурації ПЛК задати ім'я змінної, наприклад sbr. Надалі саме це ім'я ми будемо використовувати при написанні програми. Для завдання імені необхідно двічі натиснути ЛКМ на написи AT у відповідній рядку, і в розпочатому невеликому віконці надрукувати ім'я змінної (рисунок 9). Після завдання імені необхідно натиснути Enter.

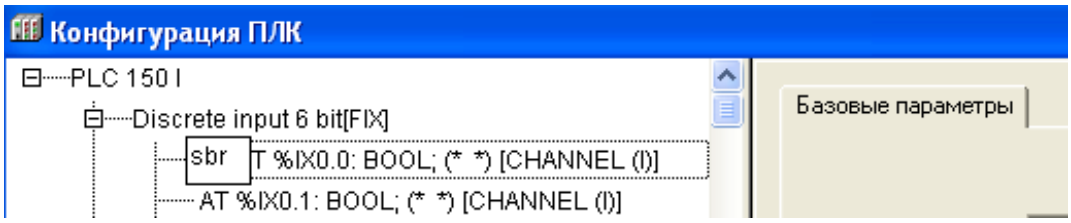

Рисунок 9 – Завдання імені змінній

Аналогічно задати решту змінних (рисунок 10).

Також потрібно налаштувати аналоговий вхід на вимірювання температури (змінна **temp**).

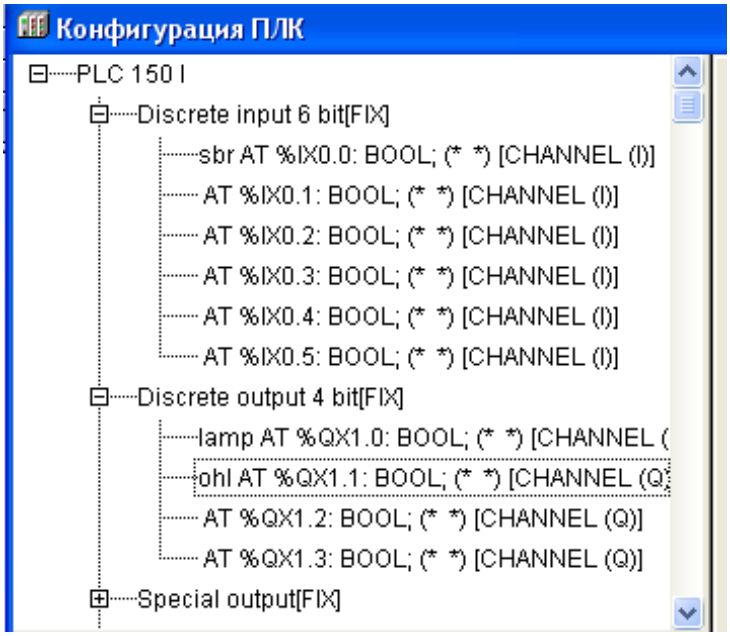

Рисунок 10 – Конфігурування каналів входів / виходів контролера

Для конфігурування аналогових входів ПЛК в його ресурсах є чотири окремих слота *Unifed signal sensor [SLOT]*. Кожен слот відповідає одній фізичній аналоговому входу ПЛК і містить два канали: власне вимірювальний (типу *REAL* з коментарем *Value*) і канал визначення часу циклу вимірювання (типу *WORD* з коментарем *Circular time*). Перший канал необхідний для зберігання поточного значення параметра, а другий - для обчислення сигналу регулювання під час реалізації ПІД-регулювання. Слоти аналогових вхідних каналів можуть бути налаштовані на обробку сигналів від джерел сигналу різного типу: уніфікованого датчика сигналу (за замовчуванням), термометра опору, термоелектричного перетворювача (термопари) і контактного датчика («сухий контакт»). Замінити тип джерела можна за допомогою контекстного меню. Для цього ЛФМ потрібно виділити потрібний слот датчика. В результаті він буде виділено рамкою з точок. Далі натиснути ПКМ і в випадаючому вікні вибрати потрібне джерело сигналу, як це показано на рисунку 11. На цьому рисунку показаний порядок заміни уніфікованого датчика сигналу на термоопір.

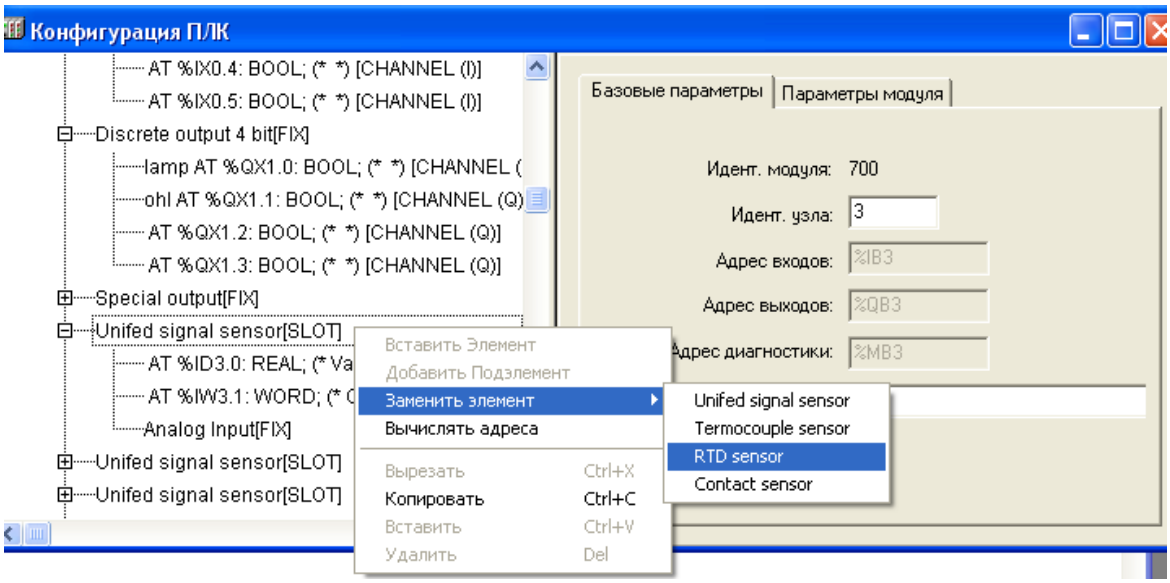

Рисунок 11 – Вибір джерела сигналу

В параметрах модуля для даного входу в полі **Type of sensor** вибрати настройку R385_500, таким чином, мається на увазі, що до контролера буде підключений датчик Pt500 (рисунок 12).

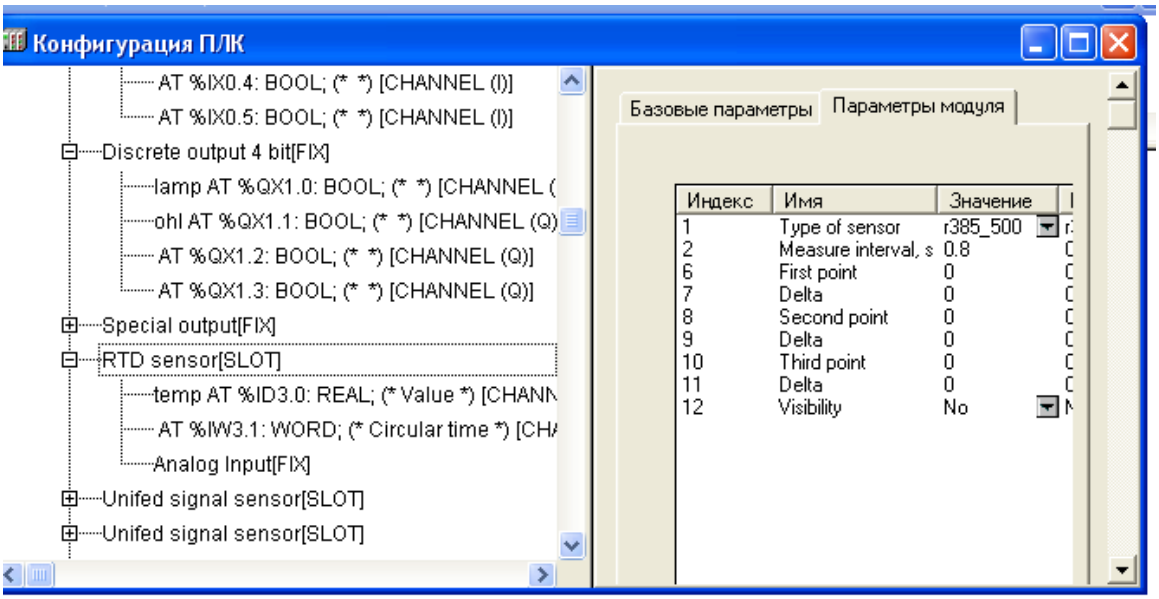

Рисунок 12 – Вибір типу датчика

Розробка програми користувача

Перейти на вкладку POU в організаторі проєктів. Створити необхідні проміжні змінні за допомогою клавіш **Shift-F2** (рисунки 13-15). Початкове значення уставки і часу спрацьовування вибрати з таблиці варіантів завдань.

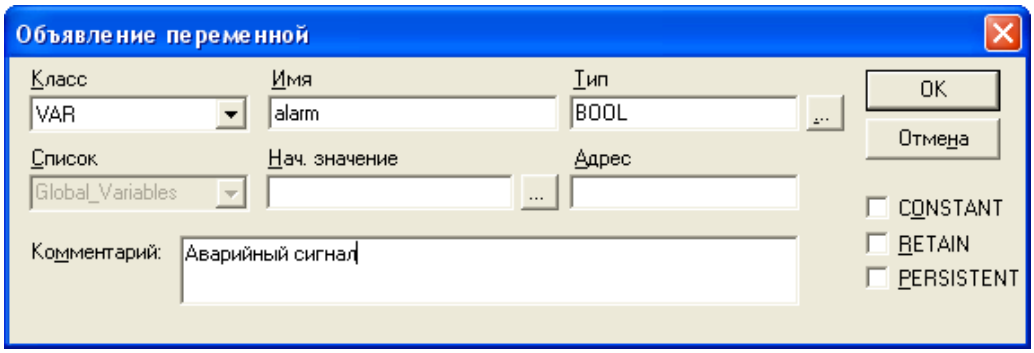

Рисунок 13 – Оголошення змінної alarm

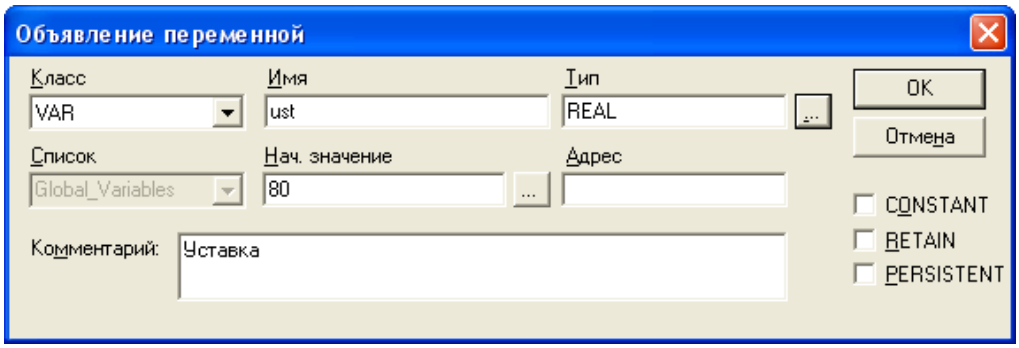

Рисунок 14 – Оголошення змінної ust

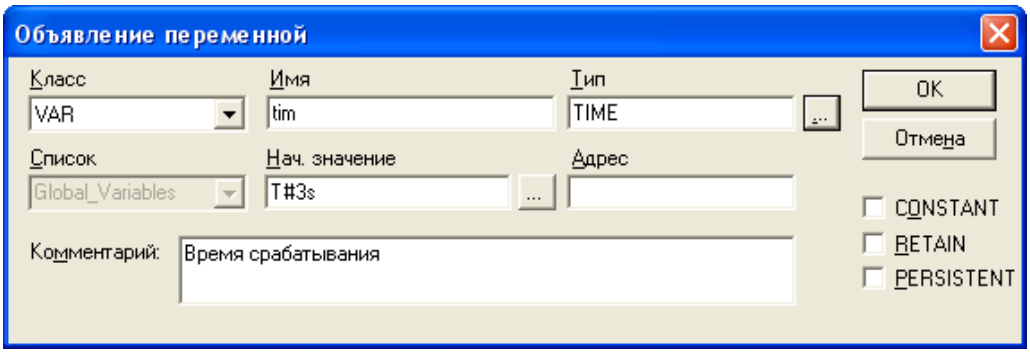

Рисунок 15 – Оголошення змінної tim

В результаті розділ оголошення змінних буде виглядати, як на рисунку 16.

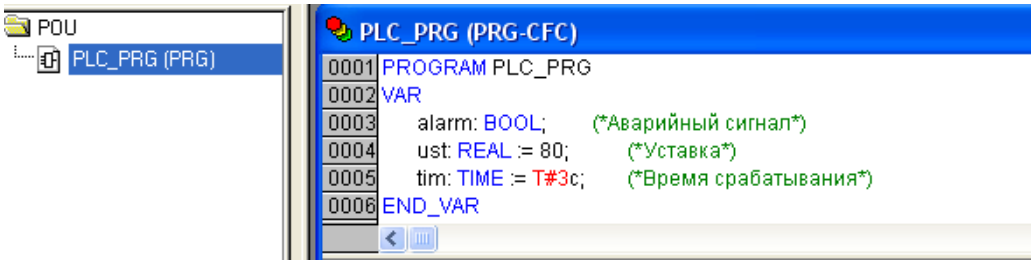

Рисунок 16 – Вікно розділу оголошення змінних

Для розробки програмного коду необхідно підключити стандартні бібліотеки: вибрати вкладку **Ресурси** - 2ЛКМ на утиліті **Менеджер бібліотек** -

меню **Вставка - Додати бібліотеку**, з'явиться вікно, в якому вибрати спочатку бібліотеку **Standart.lib**, потім **-Util.lib**.

Скласти програму на мові **CFC**, як показано на рисунку 17.

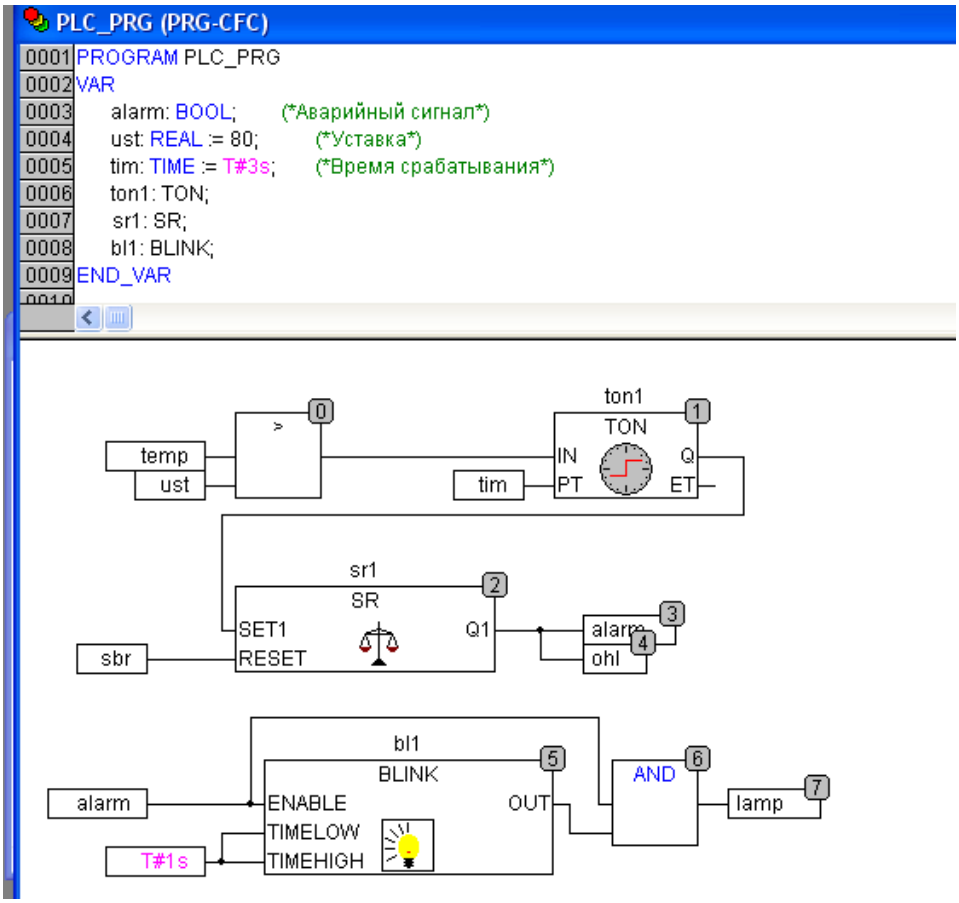

Рисунок 17 – Програма на мові CFC

Стандартні функціональні блоки вставляємо в програму за допомогою клавіші **F2**.

Зберегти проект.

Створення візуалізації

Перейти на вкладку **Візуалізація**. Натиснути ПКМ, додати об'єкт, у вікні ввести ім'я візуалізації, наприклад **plc_viz**.

Розмістити графічні елементи на екранній формі, як показано на рисунку 18.

Конфігурування елемента Повзунок, який буде змінювати температуру (рисунок 19).

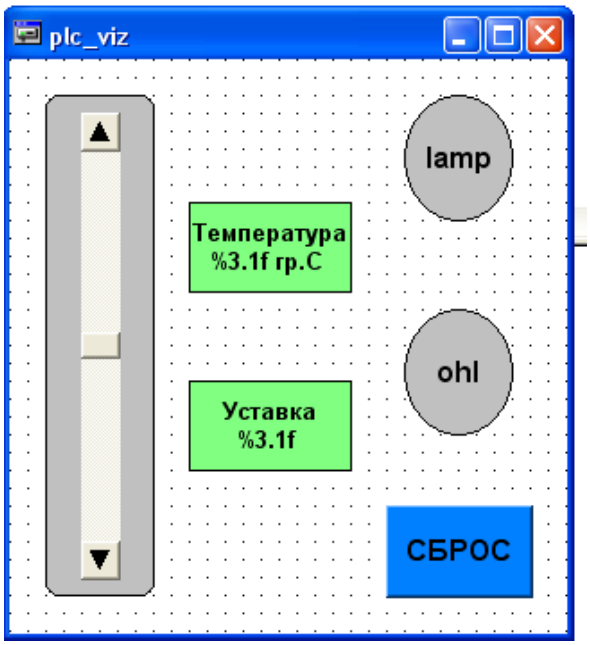

Рисунок 18 – Створення графічного екрану

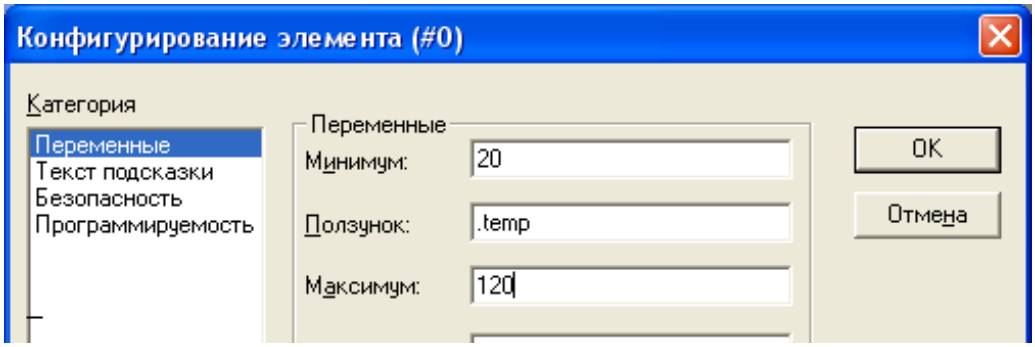

Рисунок 19 – Конфігурування елемента Повзунок

Виділити елемент **Прямокутник** під **Повзунком**, натиснути ПКМ - **На задній план** (щоб краще було видно Повзунок).

Конфігурування елемента **Прямокутник**: категорія **Форма** (рисунок 20). Категорія **Кольори**: заливка сірого кольору.

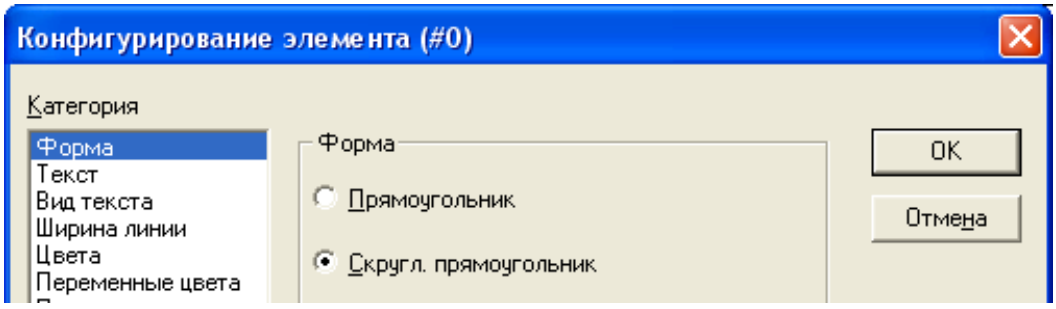

Рисунок 20 – Конфігурування елемента Прямокутник

Конфігурування елемента **Прямокутник** для виведення значення температури в числовому форматі (рисунок 21 - 22). **Кольори**: заливка - зелений.

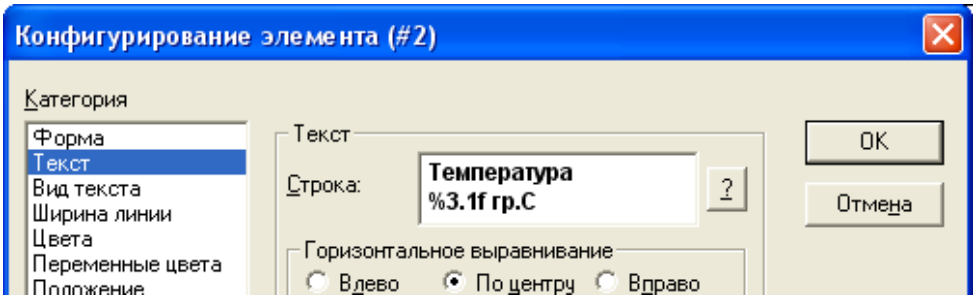

Рисунок 21 – Настройка категорії Текст для температури

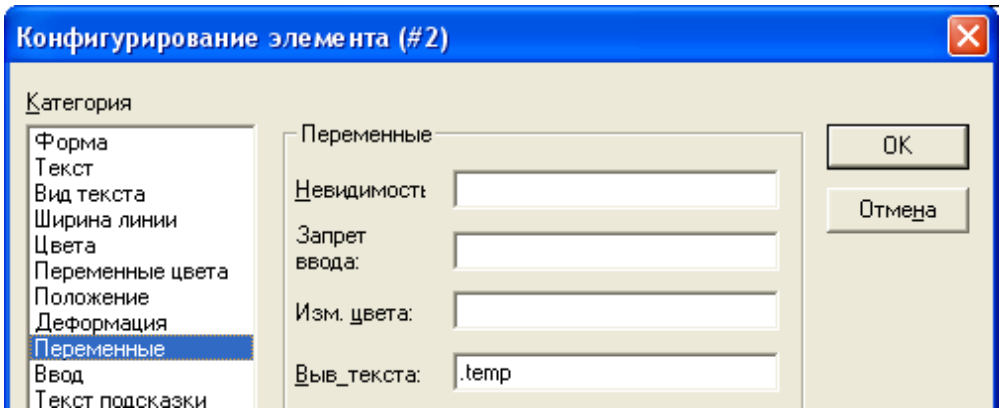

Рисунок 22 – Настройка категорії Змінні для температури

Скопіювати прямокутник з настройками і зробити копію для виведення значення уставки, відредагувати категорії, як показано на рисунках 23-25.

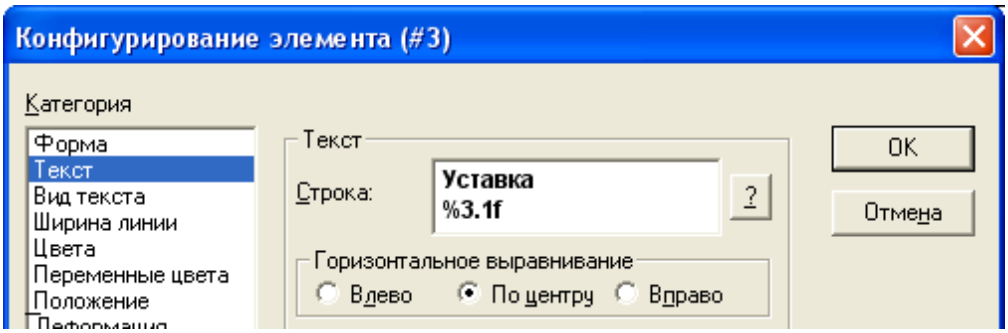

Рисунок 23 – Настройка категорії Текст для уставки

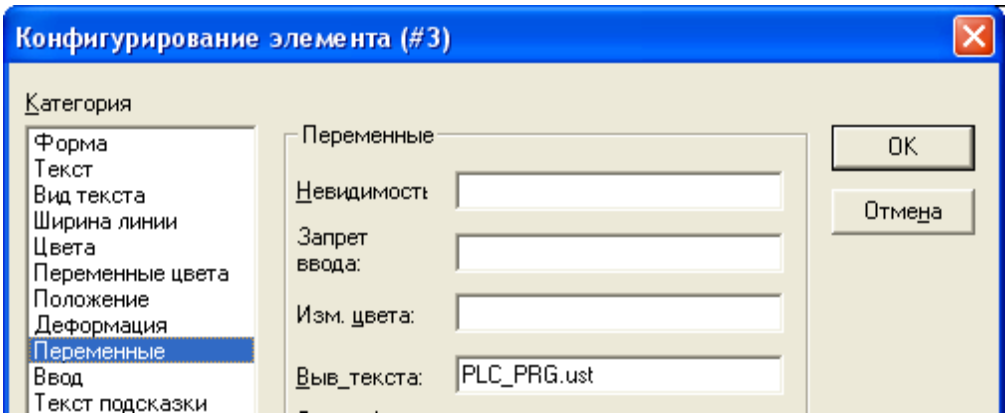

Рисунок 24 – Настройка категорії Змінні для уставки

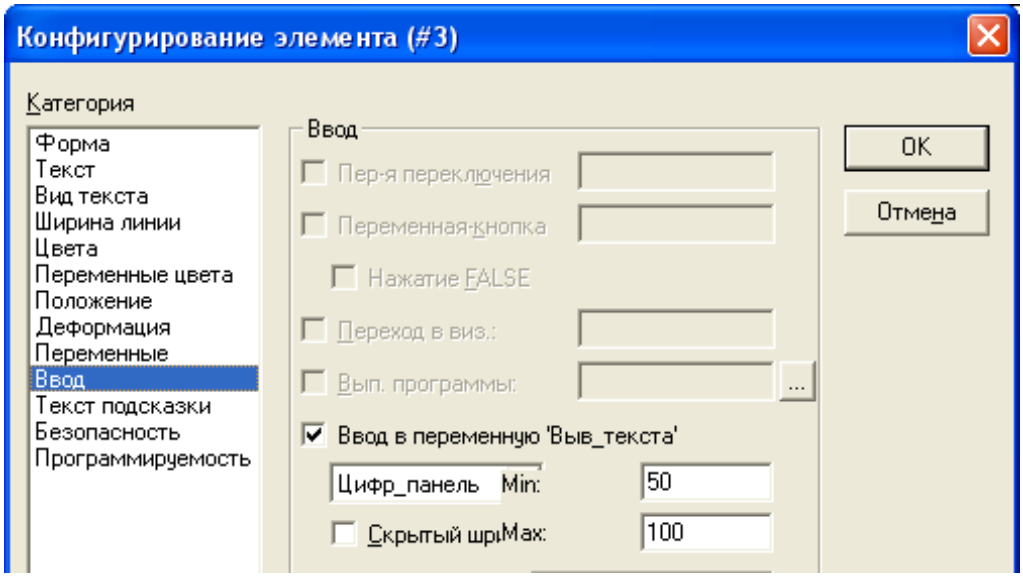

Рисунок 25 – Настройка категорії введення для уставки

Конфігурування елемента **Еліпс** для зображення лампи сигналізації (рисунки 26 –28). **Кольори**: **Заливка** - сірий колір, **Тривожний колір** - червоний.

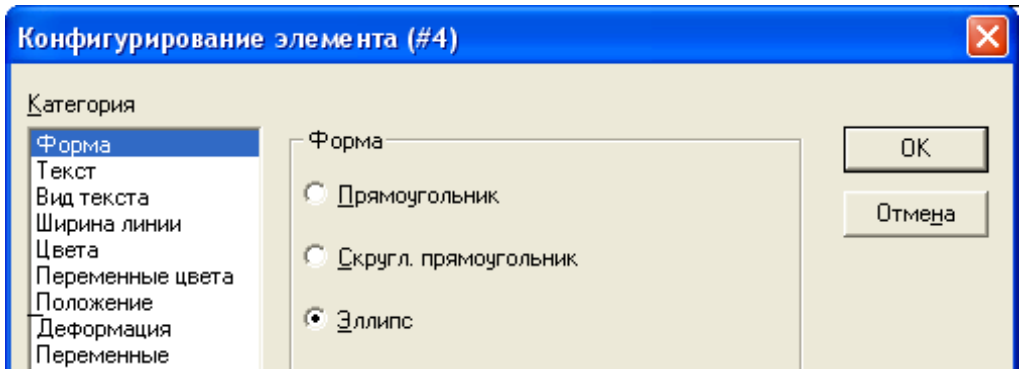

Рисунок 26 – Конфігурування елемента Еліпс

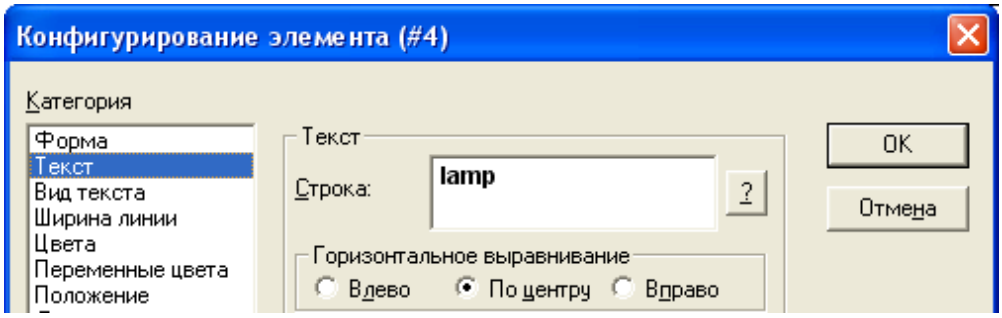

Рисунок 27 – Настройка категорії Текст для лампи

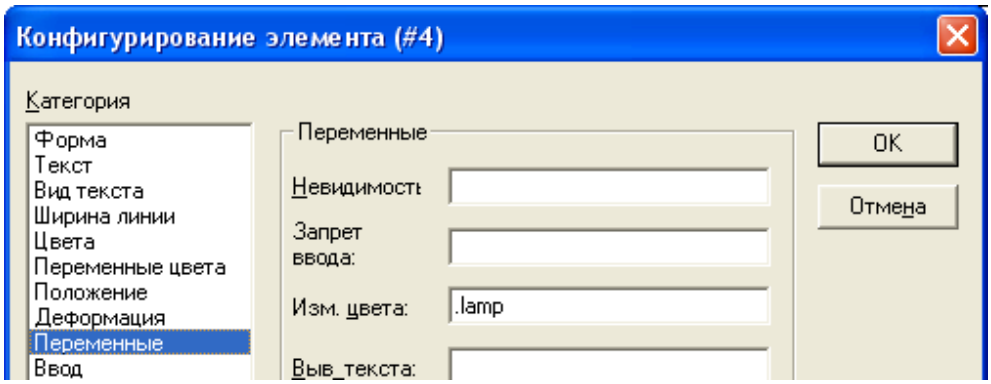

Рисунок 28 – Настройка категорії Змінні для лампи

Скопіювати еліпс з настройками і зробити копію для охолоджувача, відредагувати категорії (рисунки 29 –30). **Кольори** - ті ж самі.

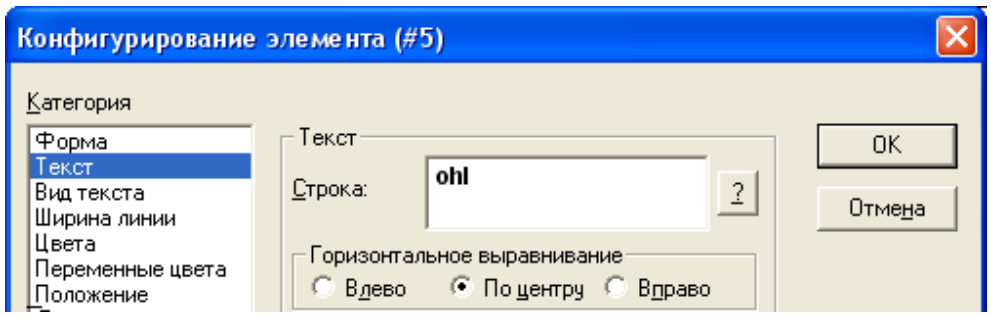

Рисунок 29 – Настройка категорії Текст для охолоджувача

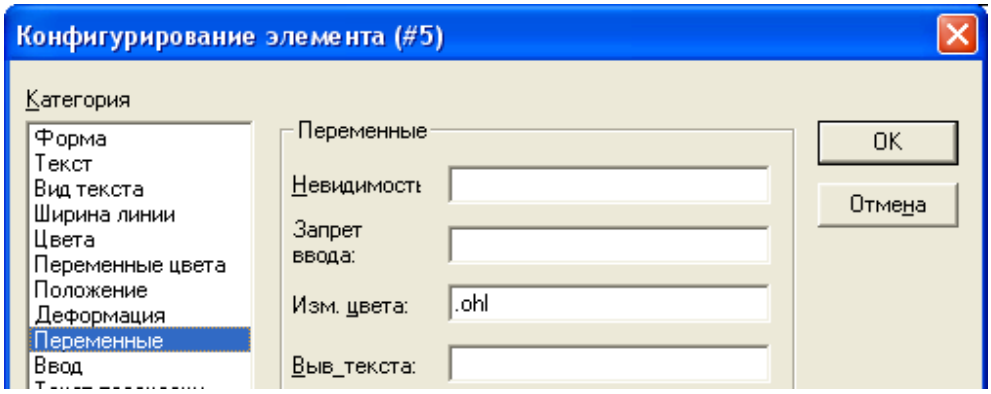

Рисунок 30 – Настройка категорії Змінні для охолоджувача

Вставити кнопку сбросу. Виконати конфігурування елемента **Кнопка** (рисунки 31 –32).

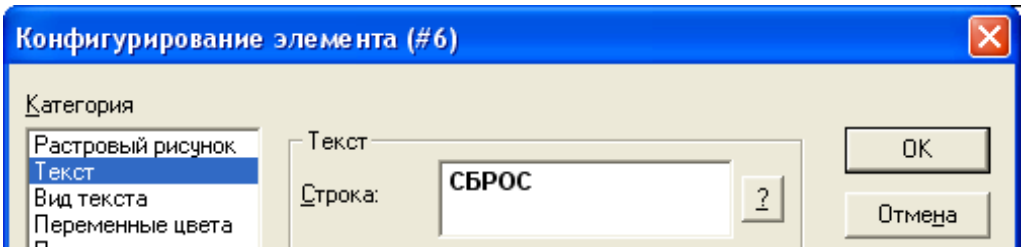

Рисунок 31 – Настройка категорії Текст кнопки СБРОС

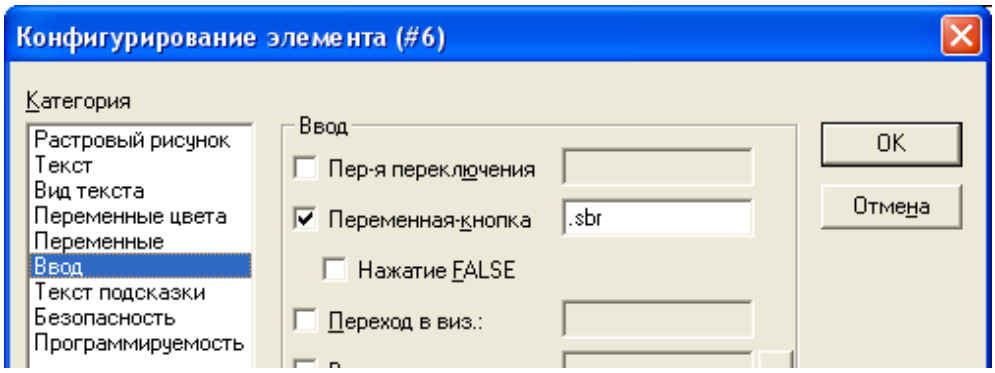

Рисунок 32 – Настройка категорії Введення кнопки СБРОС

Виконати компіляцію проєкту. В меню **Онлайн** включити **Режим емуляції**. **Підключення - Старт**. Виконати тестування проекту.

Тест № 1

Пересуваючи движок повзунка, дослідити можливі ситуації.

Дослідити нормальний режим, коли температура не перевищила уставку (рисунок 33).

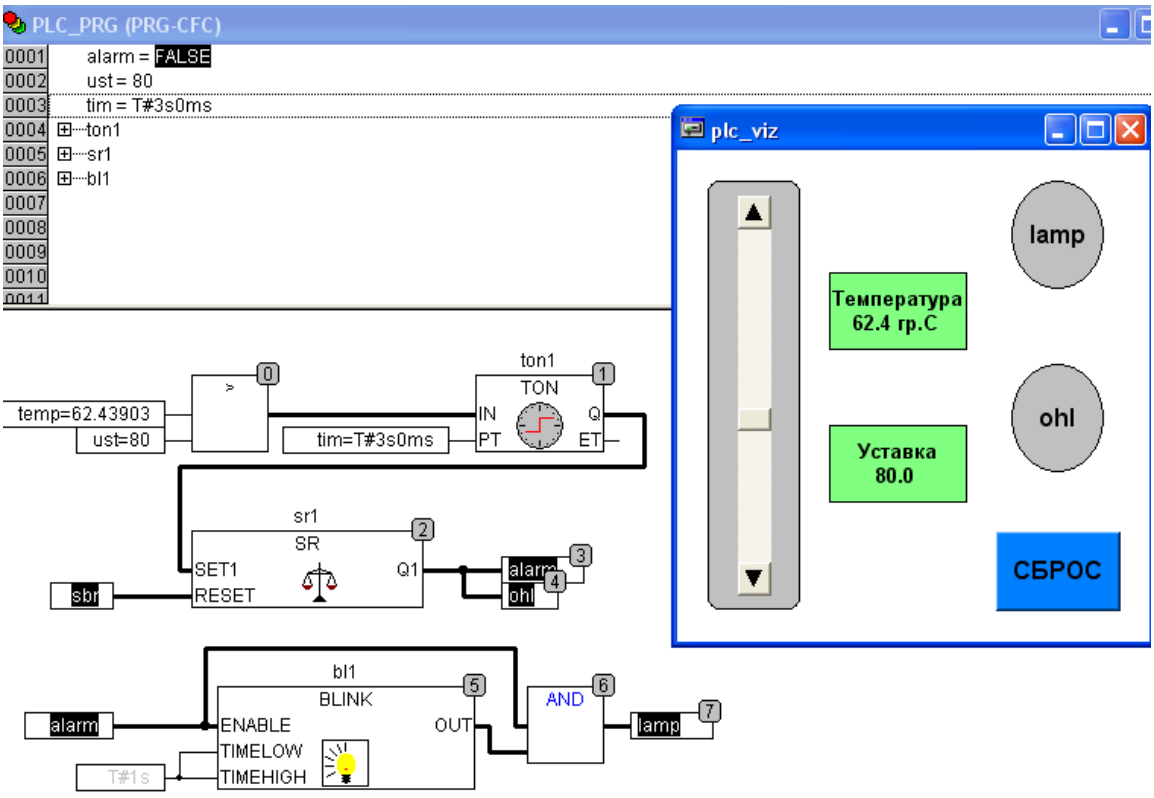

Рисунок 33 – Дослідження нормального режиму роботи установки

Встановити температуру вище уставки, через 3 сек починає блимати лампа сигналізації і включається лампа охолоджувальної системи (рисунок 34).

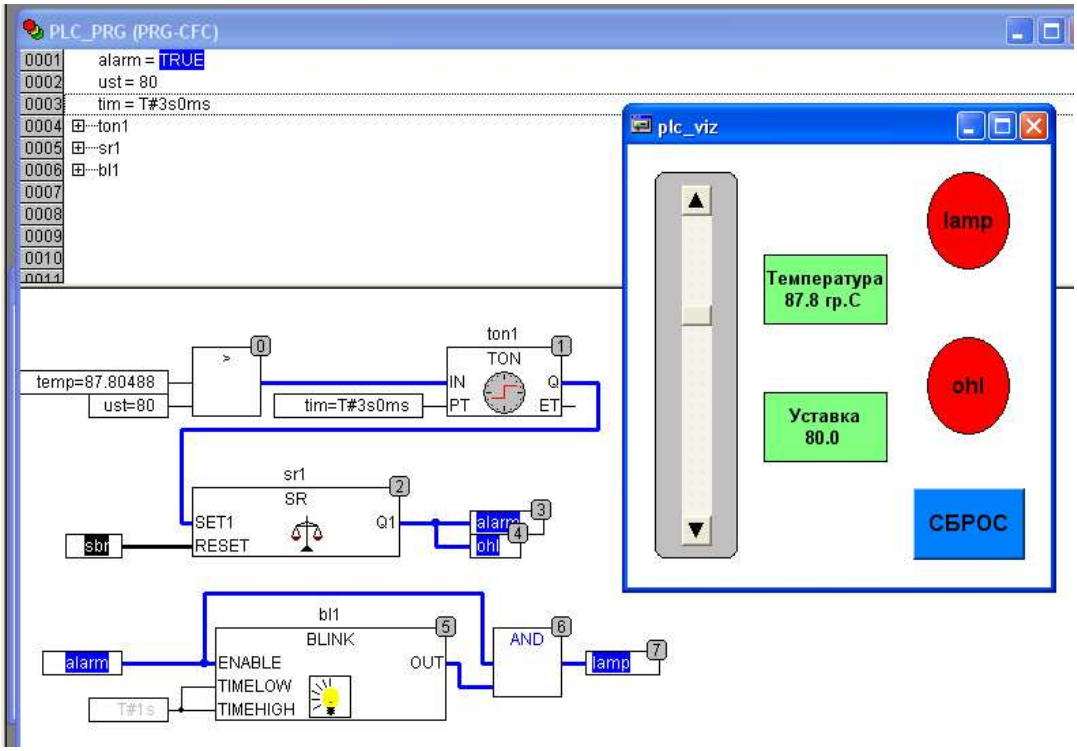

Рисунок 34 – Дослідження аварійного режиму

Повертаємо температуру в нормальний стан, але охолоджувач не вимкнено (рисунок 35).

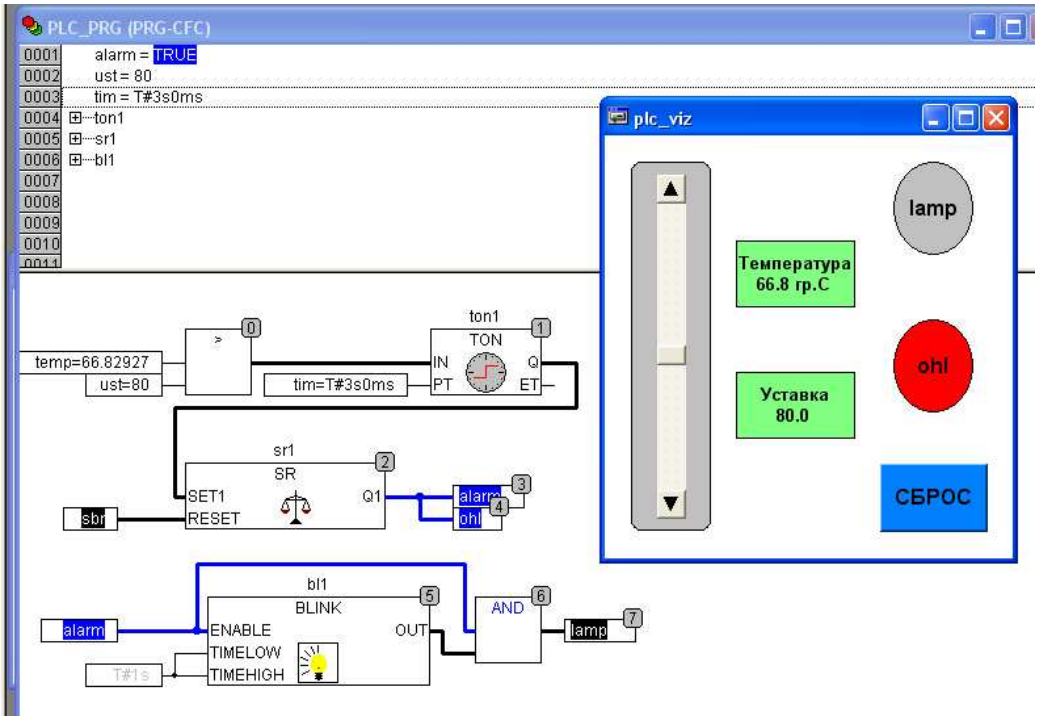

Рисунок 35 – Повернення до нормального стану

Необхідно натиснути кнопку **СБРОС**. Ситуація після натиснення кнопки **СБРОС** показана на рисунку 36.

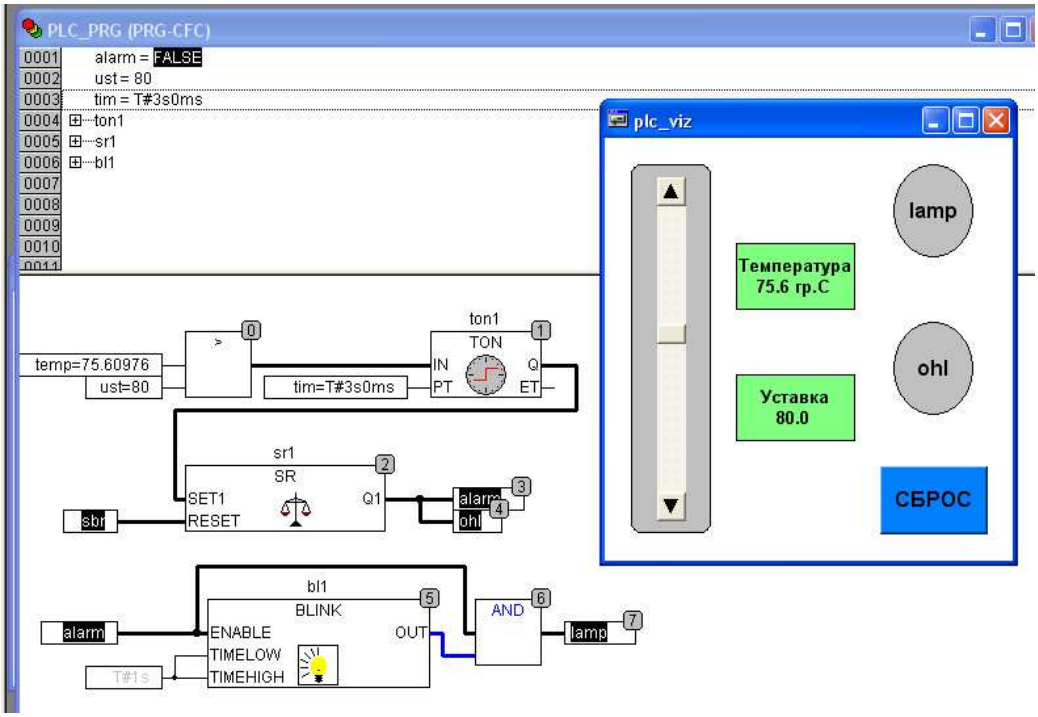

Рисунок 36 – Ситуація після натиснення кнопки СБРОС

Тест № 2

Виконати зміна уставки (з табл. 1): натиснути ЛКМ на прямокутнику Уставка, з'явиться цифрова панель, в ній ввести нове значення уставки - 60°С, натиснути ОК (рисунок 37).

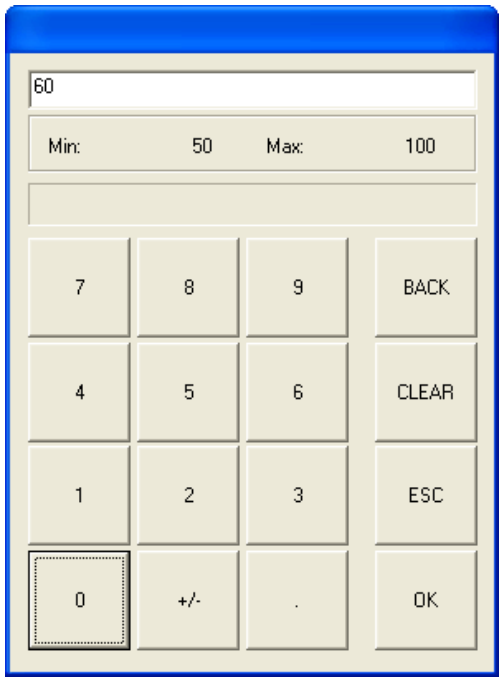

Рисунок 37 – Вікно цифрової панелі

Дослідити роботу системи в нормальному режимі (рисунок 38).

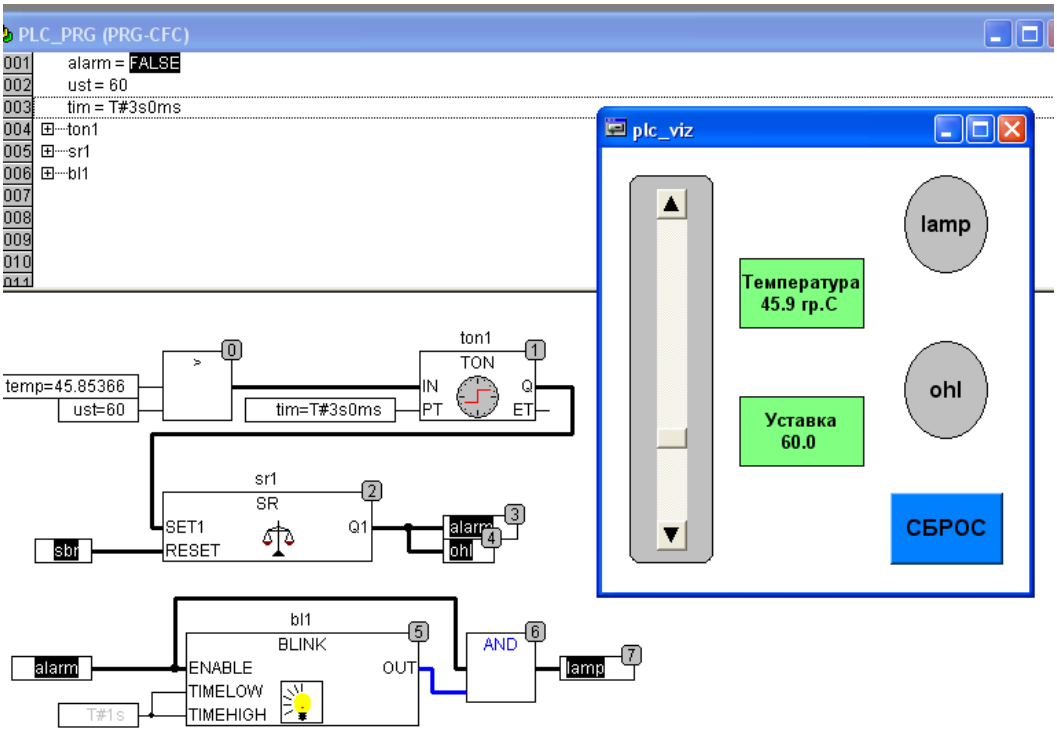

Рисунок 38 – Нормальний режим

Дослідження аварійної ситуації показано на рисунку 39.

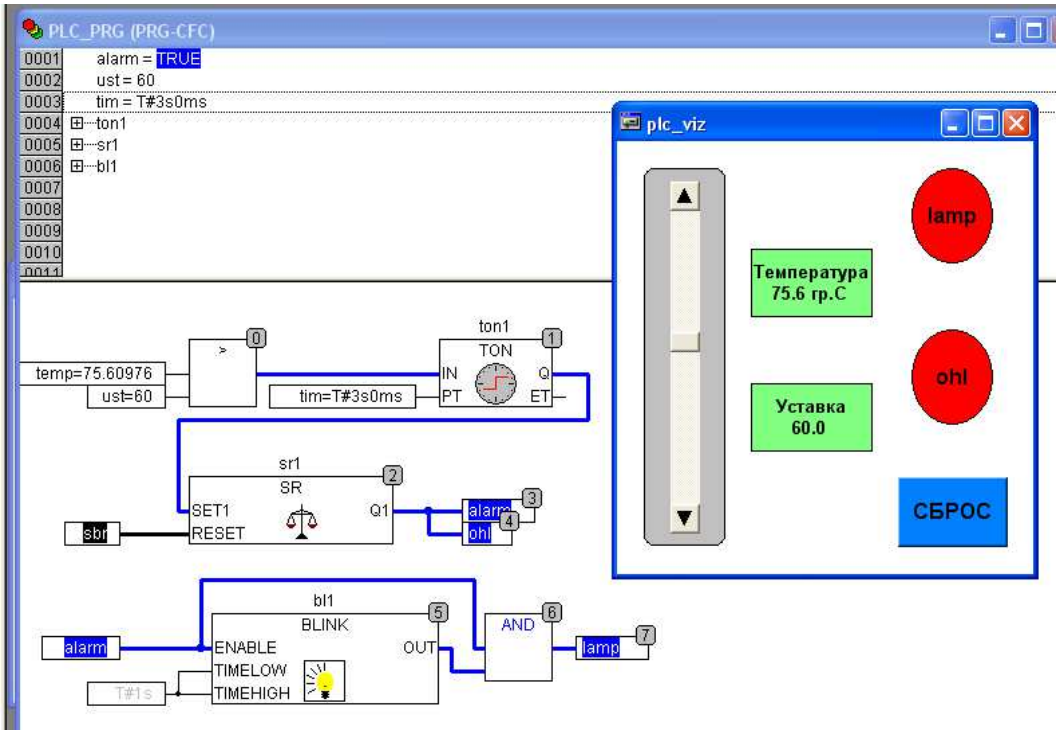

Рисунок 39 – Аварійна ситуація

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№ вар	Значення уставки		Мін. значення температури	Макс. значення температури	Час спрацьовування, сек
	Тест 1	Тест 2			
1	85	70	20	140	2
2	90	80	25	150	3
3	75	90	30	160	4
4	60	80	35	170	5
5	70	65	15	130	1
6	80	90	10	180	2
7	65	70	5	120	3
8	60	75	0	140	4
9	85	60	10	150	5
10	65	80	20	120	4
11	70	60	20	140	2
12	80	70	25	150	3
13	90	75	30	160	4
14	80	70	35	170	5
15	65	75	15	130	1
16	90	85	10	180	2
17	70	75	5	120	3
18	75	65	0	140	4
19	60	70	10	150	5
20	75	65	20	120	4

Контрольні питання

1. Яке призначення ПЛК–150?

2. Що містить таргет-файл?

3. Як установити таргет-файли?

4. Як додати таргет-файла до проєкту CoDeSys?

6. Як оголошуються змінні для входів і виходів?

7. Як позначаються входи і виходи ПЛК за адресами?

8. Як задати імена змінних для входів і виходів ПЛК?

9. Як виконати конфігурування аналогових входів ПЛК-150?

ЛАБОРАТОРНА РОБОТА № 10

Тема: РОЗРОБКА СИСТЕМИ КЕРУВАННЯ КЛАПАНОМ

Мета роботи: продовження вивчення функцій програмованого логічного контролера ПЛК150, можливостей використання його в системах контролю і управління технологічними процесами.

Постановка задачі: розробити віртуальну установку керування клапаном.

Порядок виконання роботи

Постановка задачі. Розробити віртуальну установку, в рамках якої необхідно реалізувати:

1. Плавне збільшення і зменшення ступеня відкриття клапана (**pol**) з зовнішніх кнопок (**plus** або **minus**).
1. Видачу керуючого сигналу 4-20 мА (**out**) з виходу ПЛК на клапан.
2. Відображення ступеня відкриття клапана (**pol**) у відсотках.
3. Сигналізацію про досягнення кінцевих положень (**zakr** і **otkr**).

Рішення задачі

Створення проєкту

Створити новий проєкт в системі CoDeSys. Для цього виберіть **'Файл'** - **'Створити'**.

Вибрати цільову платформу **PLC150.I-L** (рисунок 1) і підтвердити вибір кнопкою **ОК**.

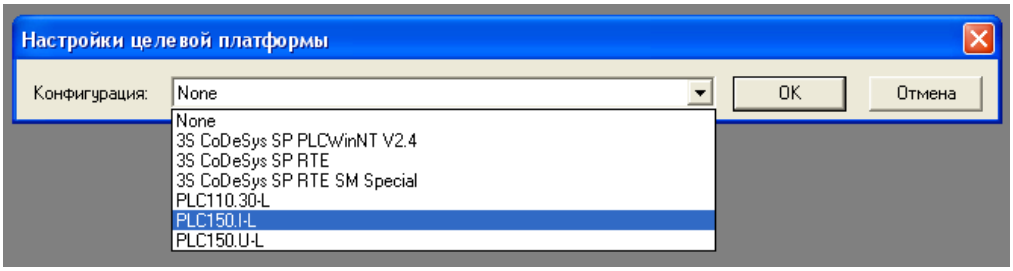

Рисунок 1 – Вибір цільової платформи ПЛК

На екрані з'явиться нове вікно, в якому будуть міститися основні параметри і настройки вибраної платформи ПЛК, натиснути кнопку «ОК» (рисунок 2).

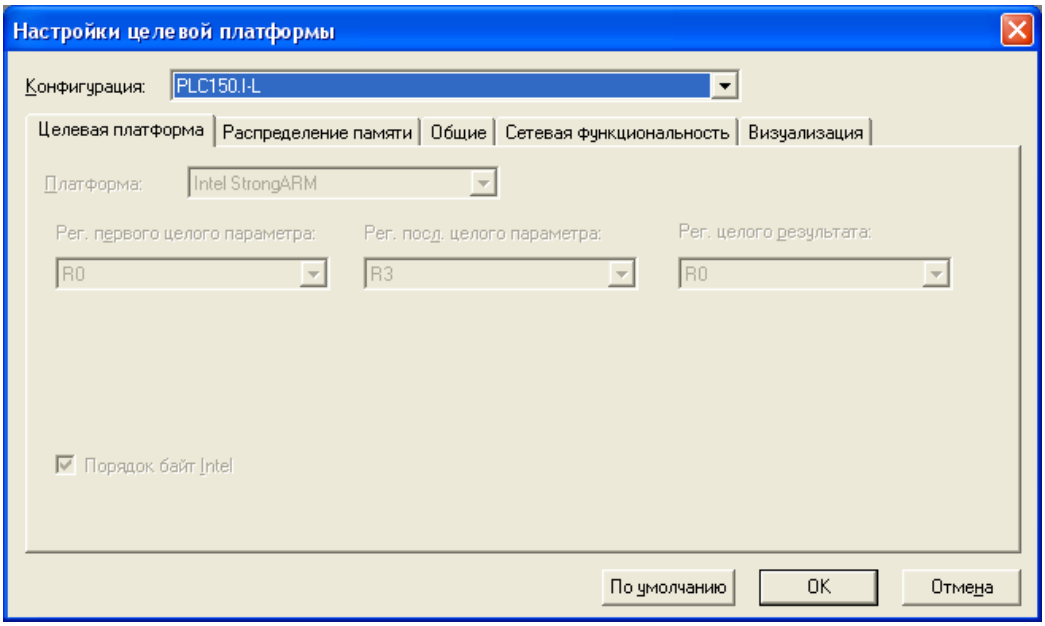

Рисунок 2 – Настройка цільової платформи ПЛК

У вікні створення нового POU його тип **Програма** і ім'я **PLC_PRG** залишити без зміни, вибрати мову реалізації **CFC**, натиснути **ОК**.

Конфігурування ПЛК

Перейти на вкладку **Ресурси**, вибрати утиліту **Конфігурація ПЛК**. На вкладці **Параметри модуля** задати час циклу - 10 мс. Далі натисканням миші по знаку «+», який знаходиться зліва поряд зі слотом *PLC150.I-L*, відкрийте його вміст. З'явиться дерево з фіксованим набором ресурсів ПЛК (рисунок 3).

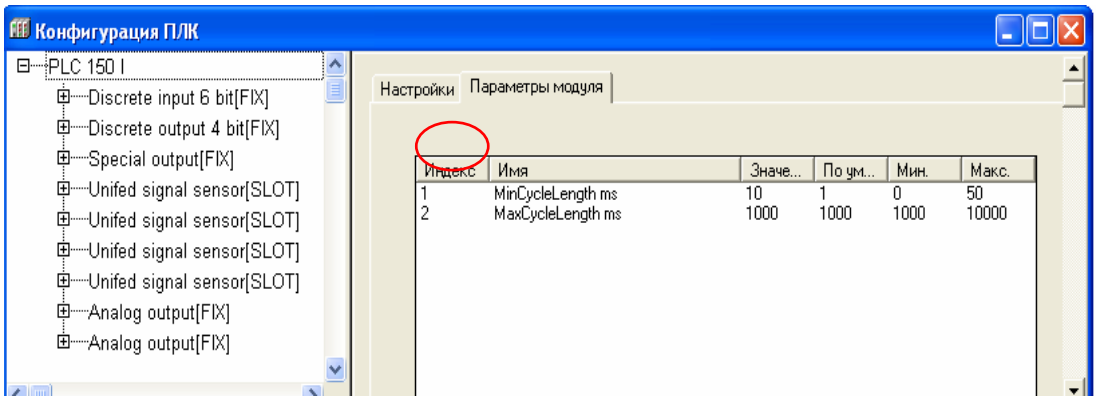

Рисунок 3 – Завдання параметрів модуля

Зберегти проект.

Для вирішення поставленого завдання необхідно використати дві дискретні вхідні змінні **plus** і **minus**, дві дискретні вихідні змінні **zakr** і **otkr**, а також вихідну аналогову змінну **out** (рисунок 4). Після завдання кожного імені необхідно натискати **Enter**.

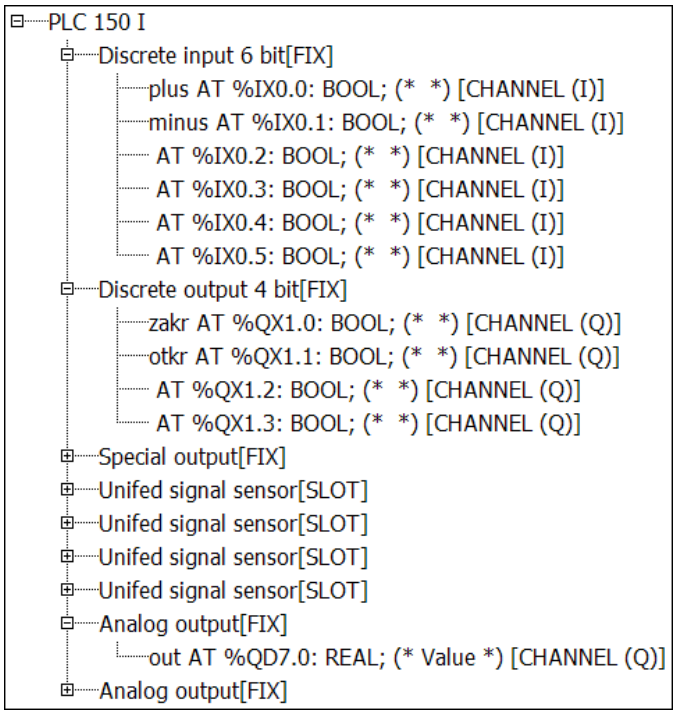

Рисунок 4 – Конфігурація каналів входів/виходів контролера

Розробка програми користувача

Перейти на вкладку POU в організаторові проектів. Створити проміжну змінну за допомогою клавіш **Shift - F2** (рисунок 5).

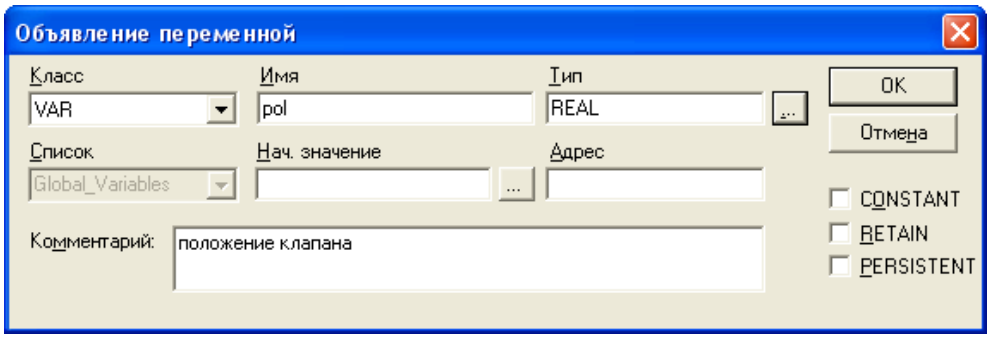

Рисунок 5 – Оголошення змінної pol

В результаті розділ оголошення змінних виглядатиме, як на рисунку 6.

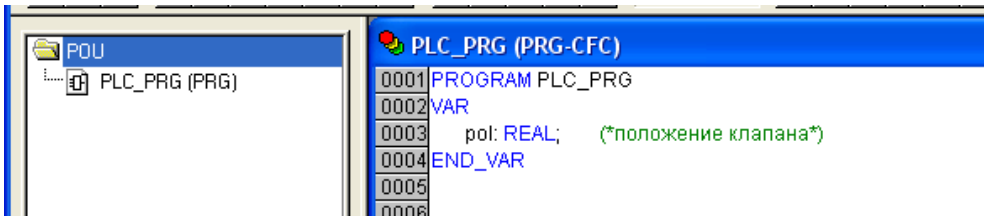

Рисунок 6 – Вікно розділу оголошення змінних

Скласти програму на мові **CFC**, як показано на рисунку 7.

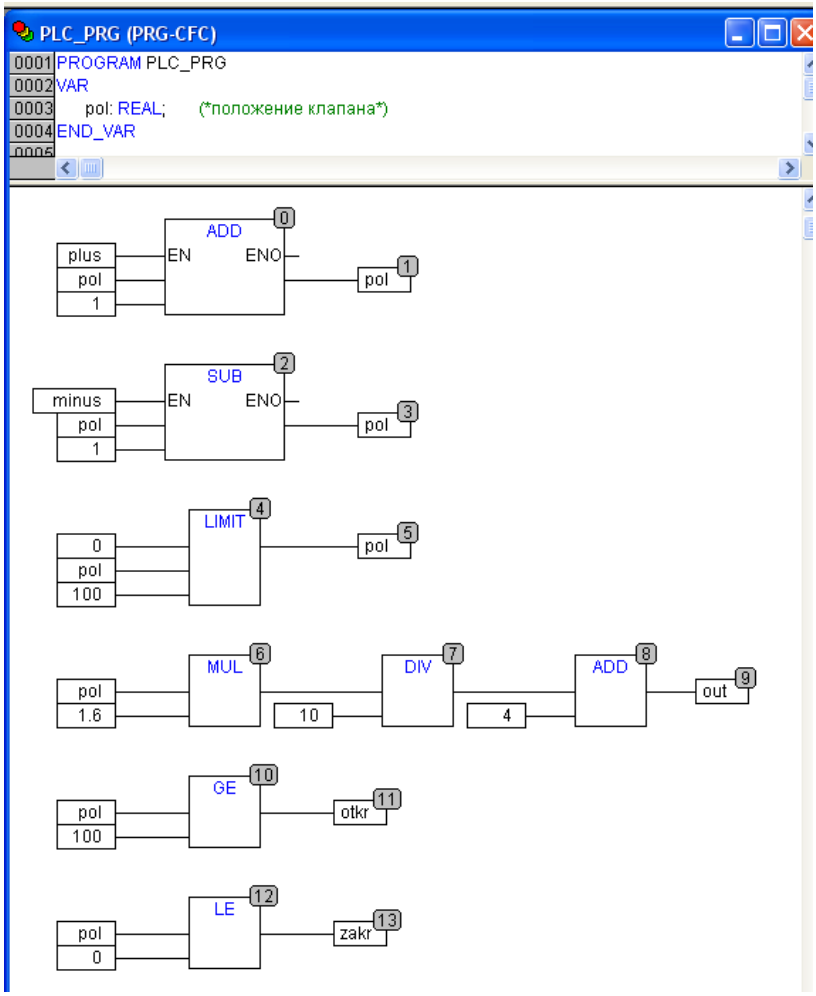

Рисунок 7 – Програма на мові CFC

Візуалізація

Виконати візуалізацію проєкту, як показано на рисунку 8.

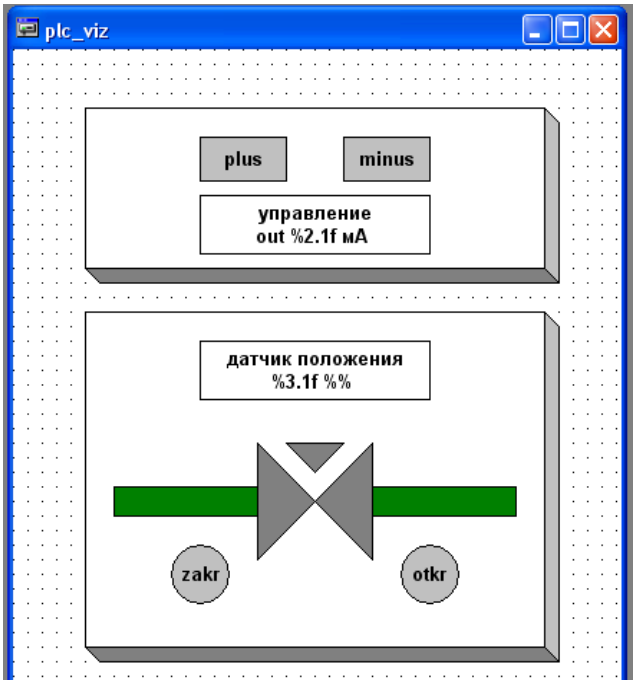

Рисунок 8 – Візуалізація проєкту

Виконати конфігурування прямокутника **plus** (рисунок 9).

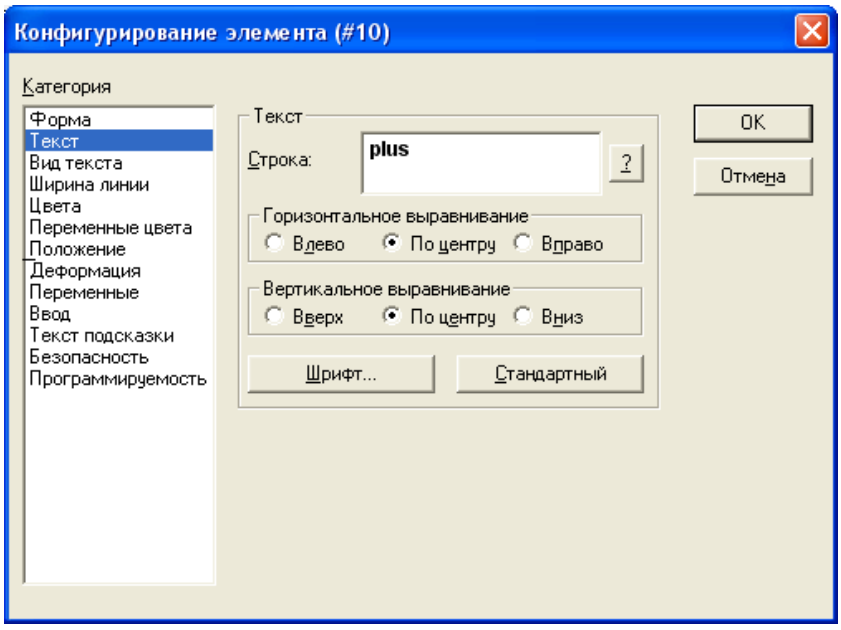

Рисунок 9 – Налаштування категорії Текст

Задати кольори: Кольори - Заливка –сірий; Тривожний колір - Заливка – Зелений.

Налаштувати категорію Змінні (рисунок 10) і категорію Введення (рисунок 11).

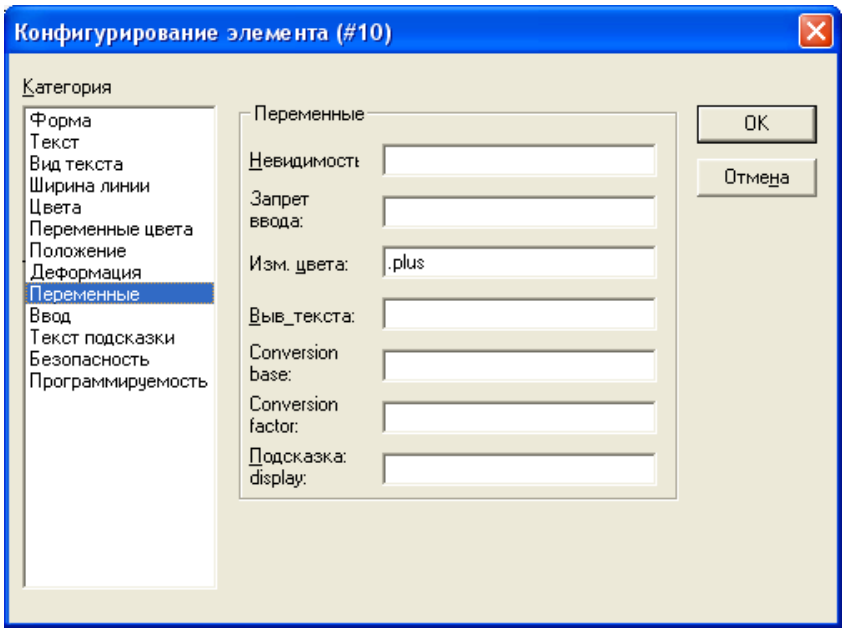

Рисунок 10 – Налаштування категорії Змінні

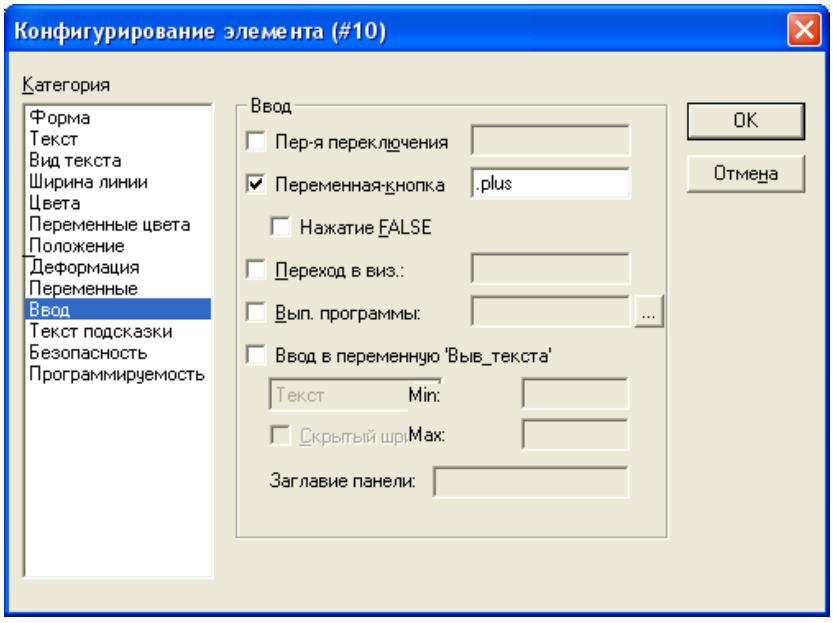

Рисунок 11 – Налаштування категорії Введення

Виконати конфігурування прямокутника **minus** (рисунок 12). Налаштувати категорію Текст.

Задати кольори: Кольори - Заливка – сірий; Тривожний колір - Заливка – Зелений.

Налаштувати категорію Змінні (рисунок 13).

Налаштувати категорію Введення (рисунок 14).

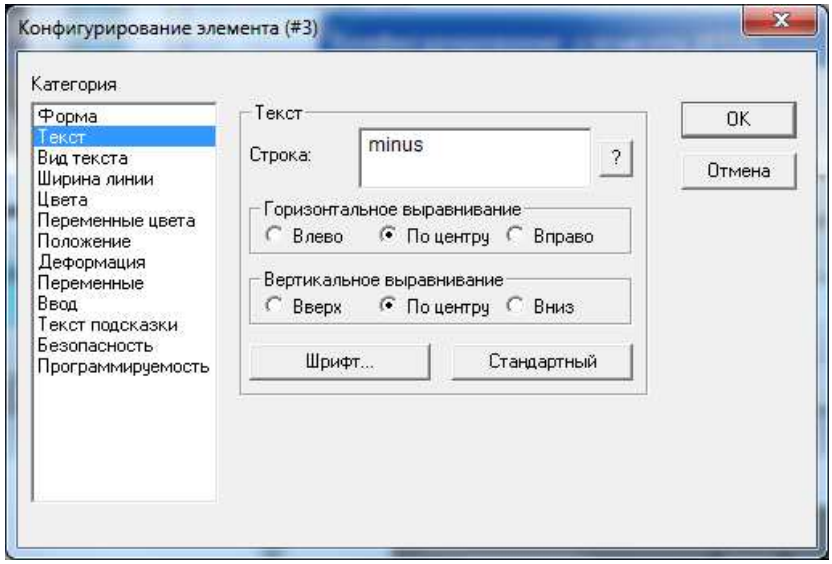

Рисунок 12 – Налаштування категорії Текст

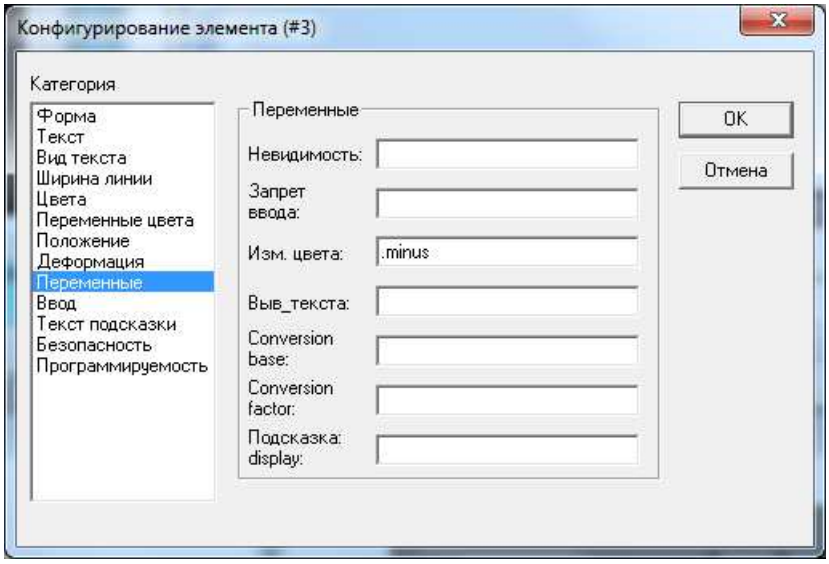

Рисунок 13 – Налаштування категорії Змінні

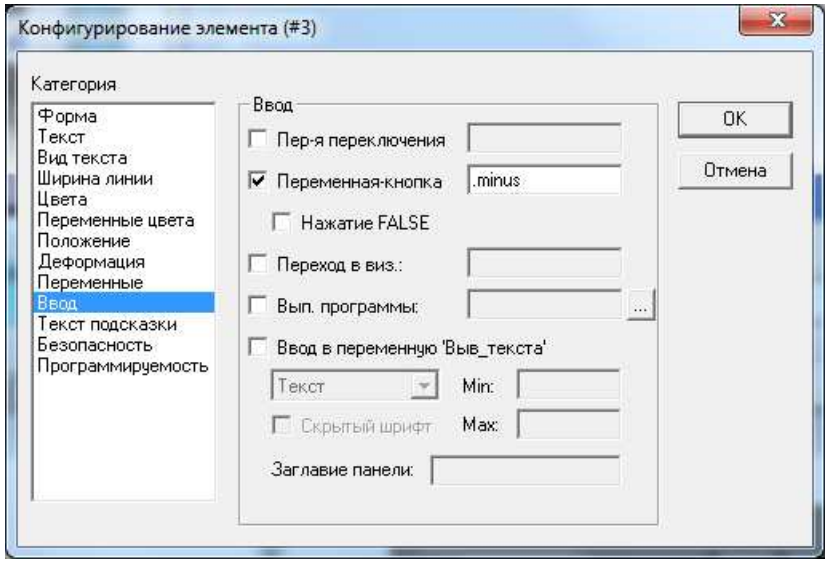

Рисунок 14 – Налаштування категорії Введення

Виконати конфігурування елемента **керування.** Налаштувати категорію Текст (рисунок 15).

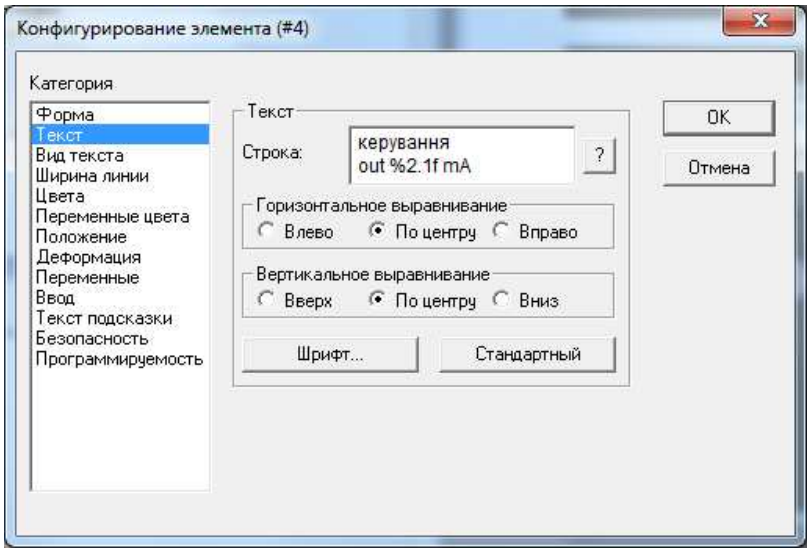

Рисунок 15 – Налаштування категорії Текст

Задати кольори: Кольори - Заливка – білий; Тривожний колір - Заливка – білий.

Налаштувати категорію Змінні (16).

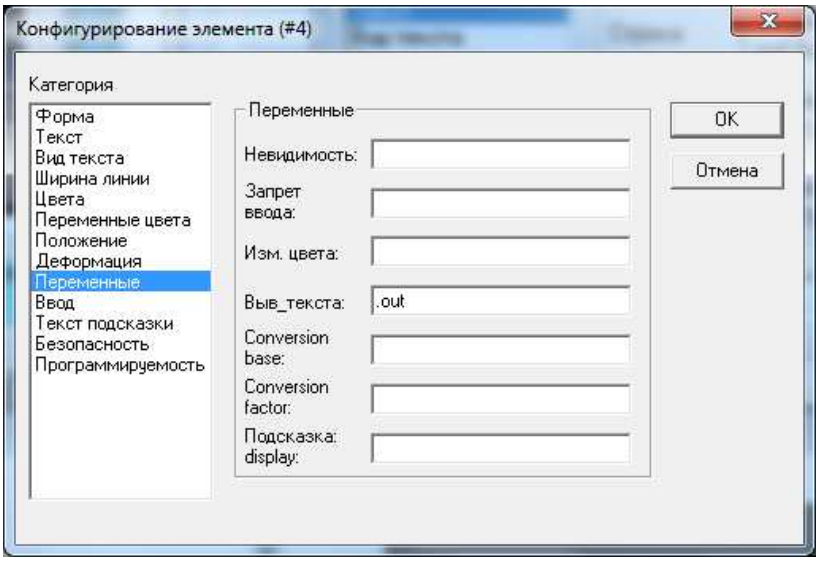

Рисунок 16 – Налаштування категорії Змінні

Виконати конфігурування елемента **датчик положення.** Налаштувати категорію Текст (рисунок 17).

Задати кольори: Кольори - Заливка – білий, Тривожний колір - Заливка – білий.

Налаштувати категорію Змінні (рисунок 18).

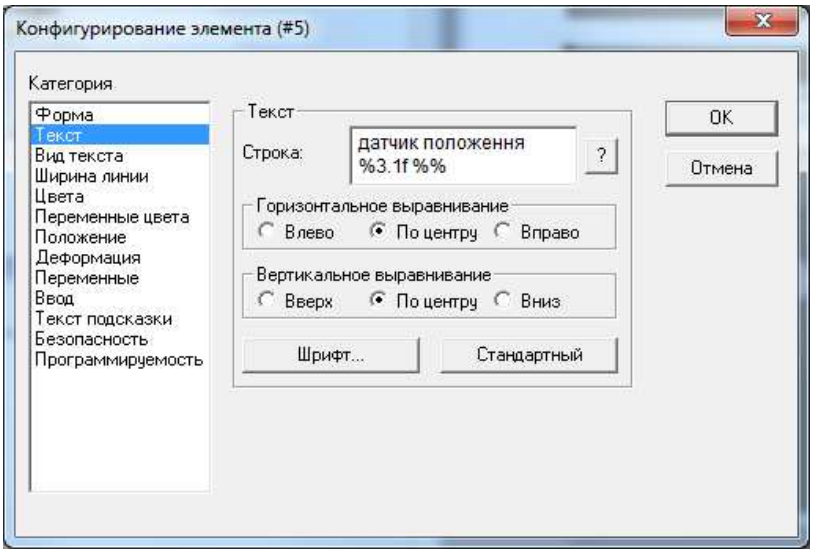

Рисунок 17 – Налаштування категорії Текст

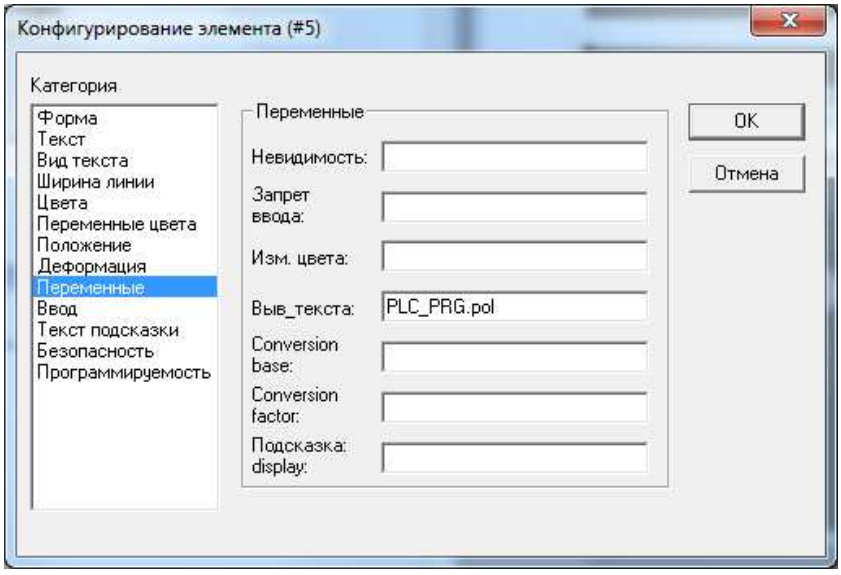

Рисунок 18 – Налаштування категорії Змінні

Побудувати два трубопроводи – прямокутники. Задати кольори: Кольори - Заливка – темнозелений, Тривожний колір - Заливка – білий.

Побудувати клапан - три полігони, налаштування кольорів для усіх однакове: Кольори - Заливка – темносірий, Тривожний колір - Заливка – зелений.

Налаштувати елемент - полігон згори, категорію Змінні (рисунок 19).

Виконати конфігурування полігона ліворуч категорію Змінні (рисунок 20).

Виконати конфігурування полігона справа категорію Змінні (рисунок 21).

Виконати конфігурування елемента: лівий еліпс. Налаштувати категорію Текст (рисунок 22).

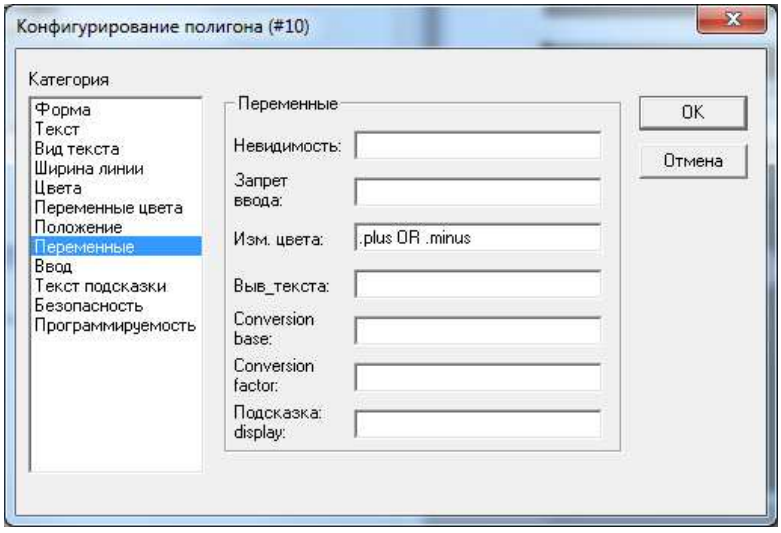

Рисунок 19 – Налаштування категорії Змінні

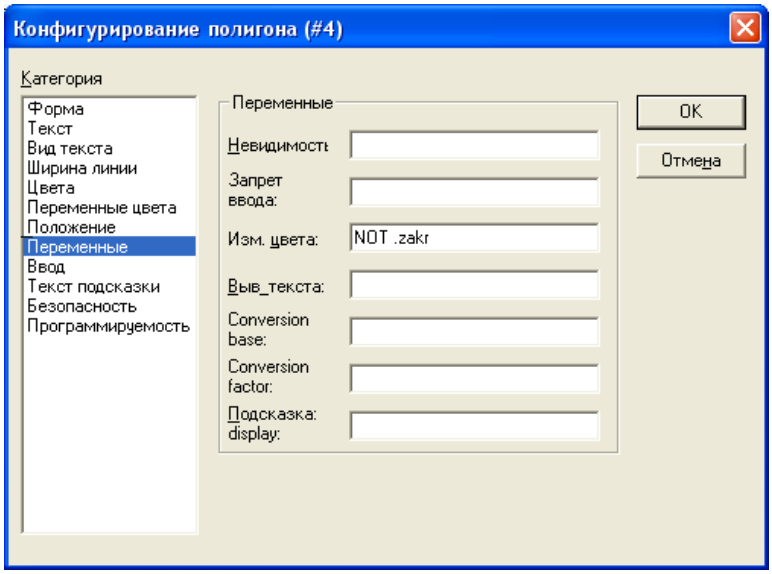

Рисунок 20 – Налаштування категорії Змінні полігона ліворуч

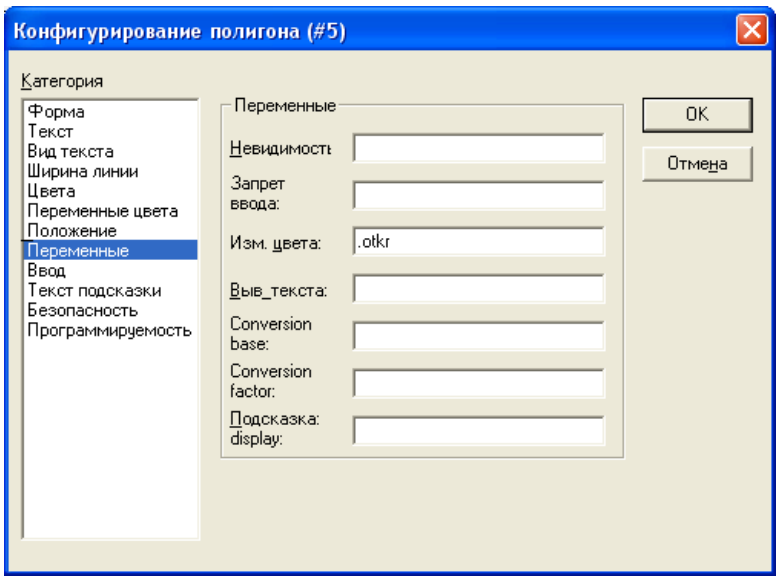

Рисунок 21 – Налаштування категорії Змінні полігона справа

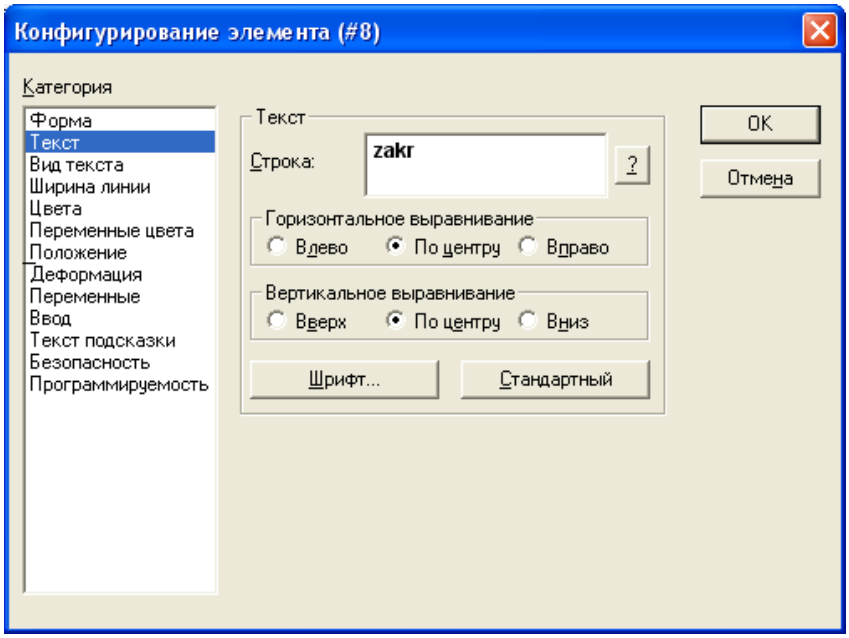

Рисунок 22 – Налаштування категорії Текст лівого еліпса

Задати кольори: Кольори - Заливка – сірий, Тривожний колір - Заливка – червоний. Налаштувати категорію Змінні (рисунок 23).

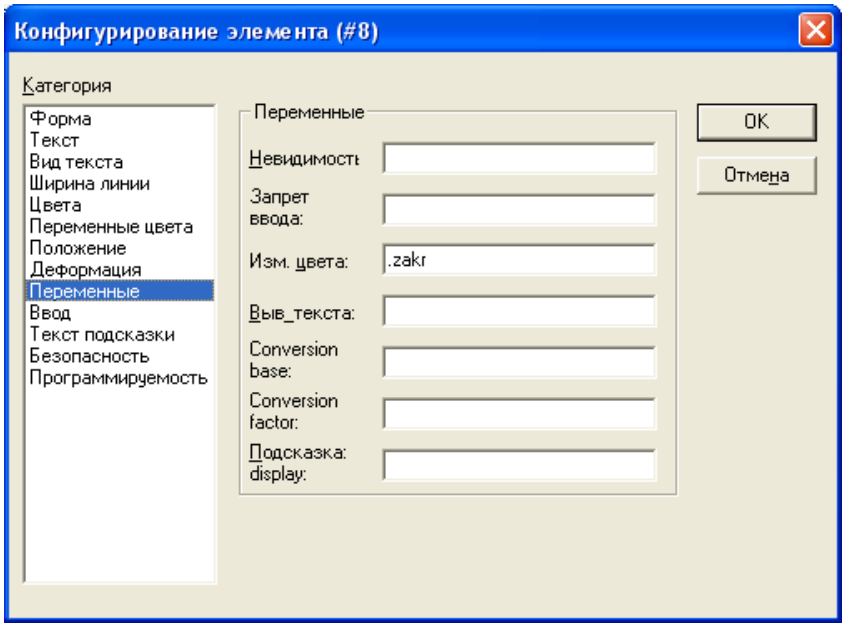

Рисунок 23 – Налаштування категорії Змінні лівого еліпса

Виконати конфігурування елемента: правий еліпс. Налаштувати категорію Текст (рисунок 24).

Задати кольори: Кольори - Заливка – сірий, Тривожний колір - Заливка – червоний.

Налаштувати категорію Змінні (рисунок 25).

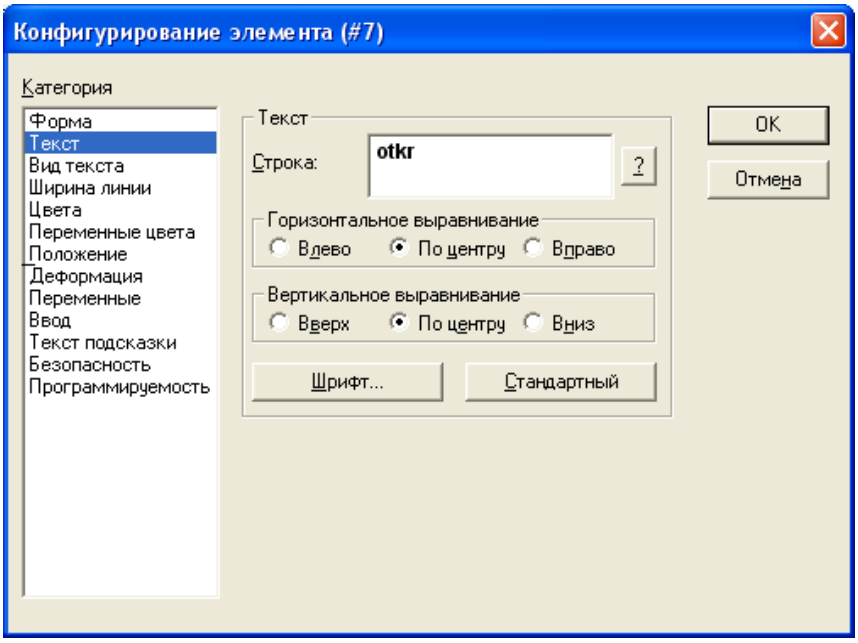

Рисунок 24 – Налаштування категорії Текст правого еліпса

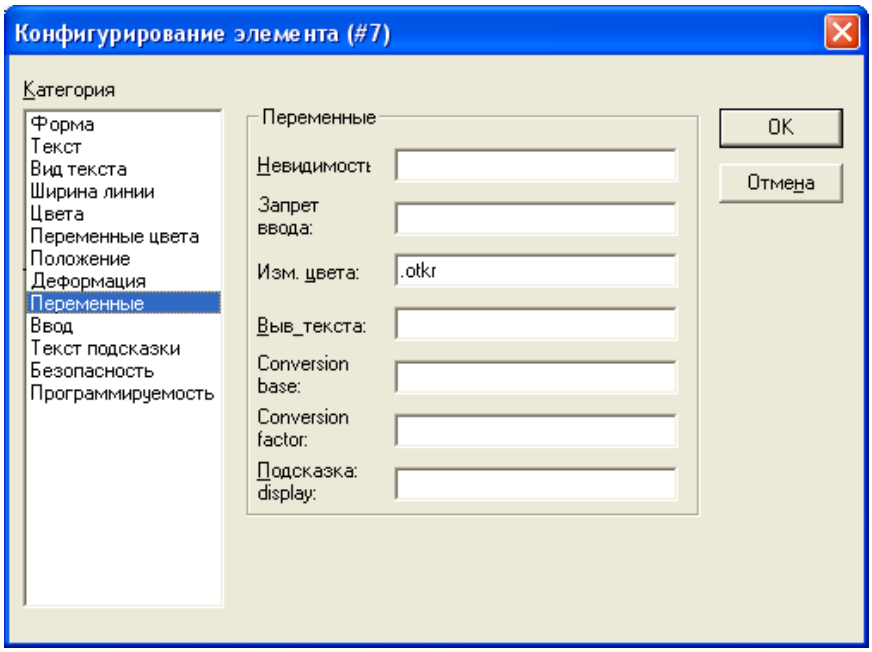

Рисунок 25 – Налаштування категорії Змінні правого еліпса

Виконати тестування проєкту.

Тест № 1

Задати нульове положення датчика (рисунок 26).

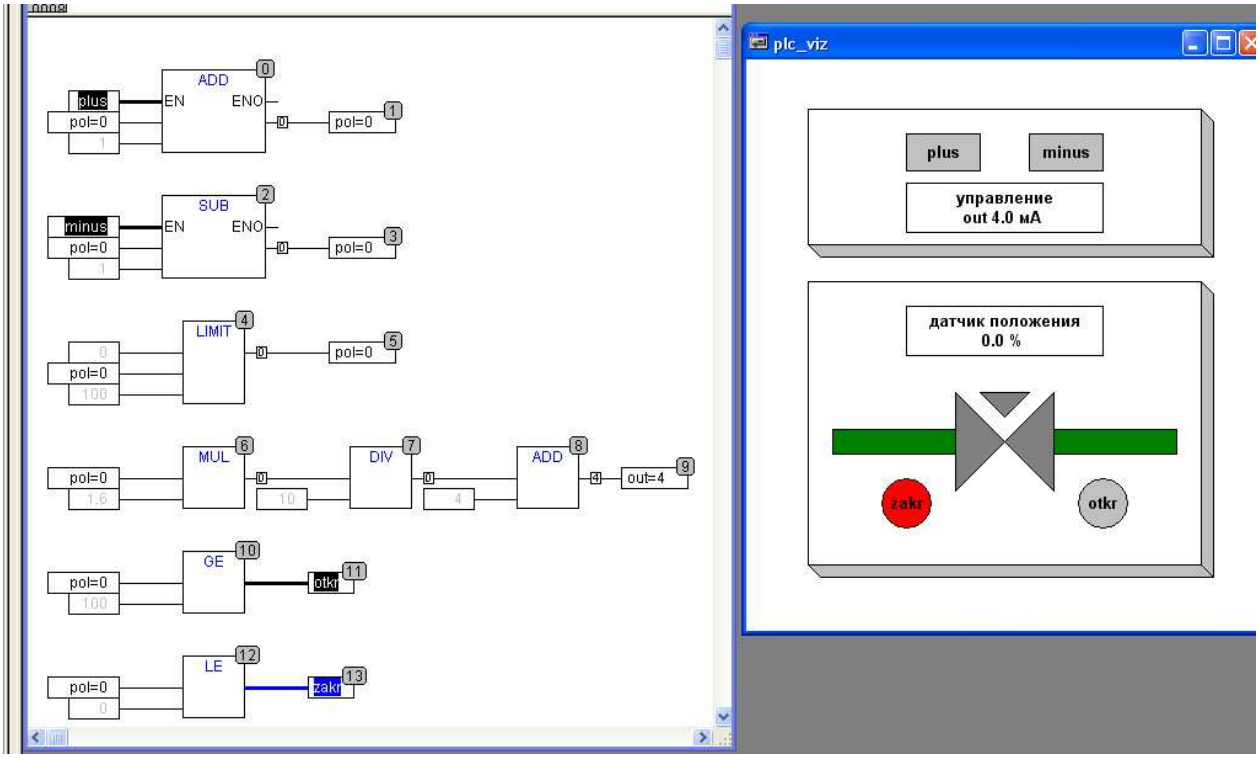

Рисунок 26 – Программа і візуалізація проєкту

Тест № 2

Відкрити клапан, натискаючи кнопку plus (рисунки 27 - 28).

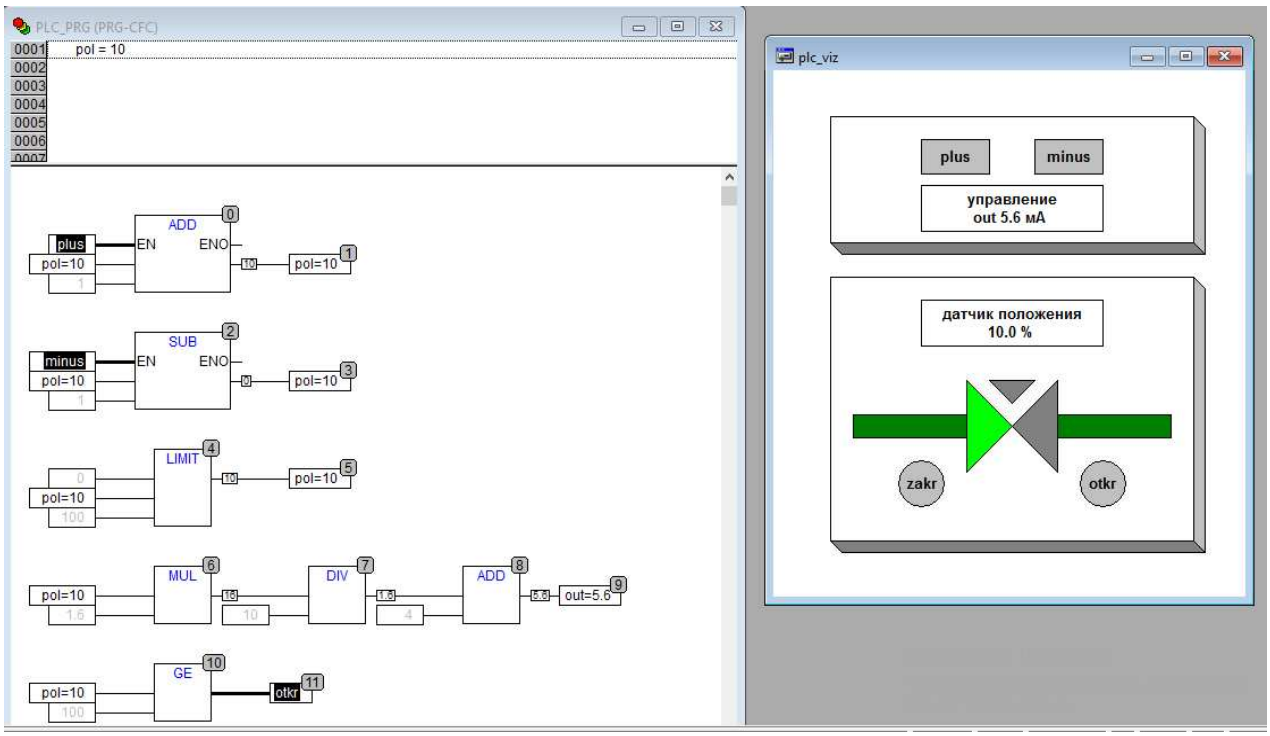

Рисунок 27 – Візуалізація проєкту

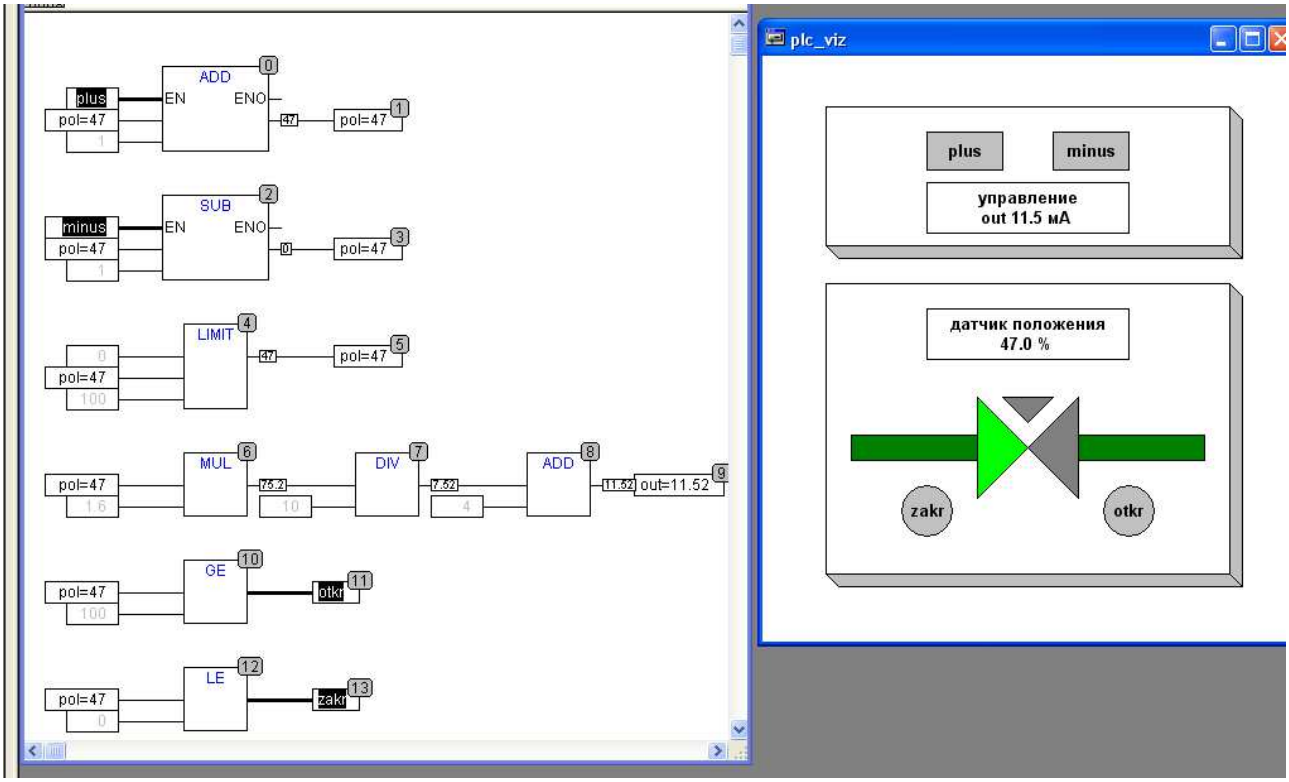

Рисунок 28 – Візуалізація проєкту

Тест № 3

Відкрити клапан повністю, при цьому загорається лампочка otkr (рисунок 29).

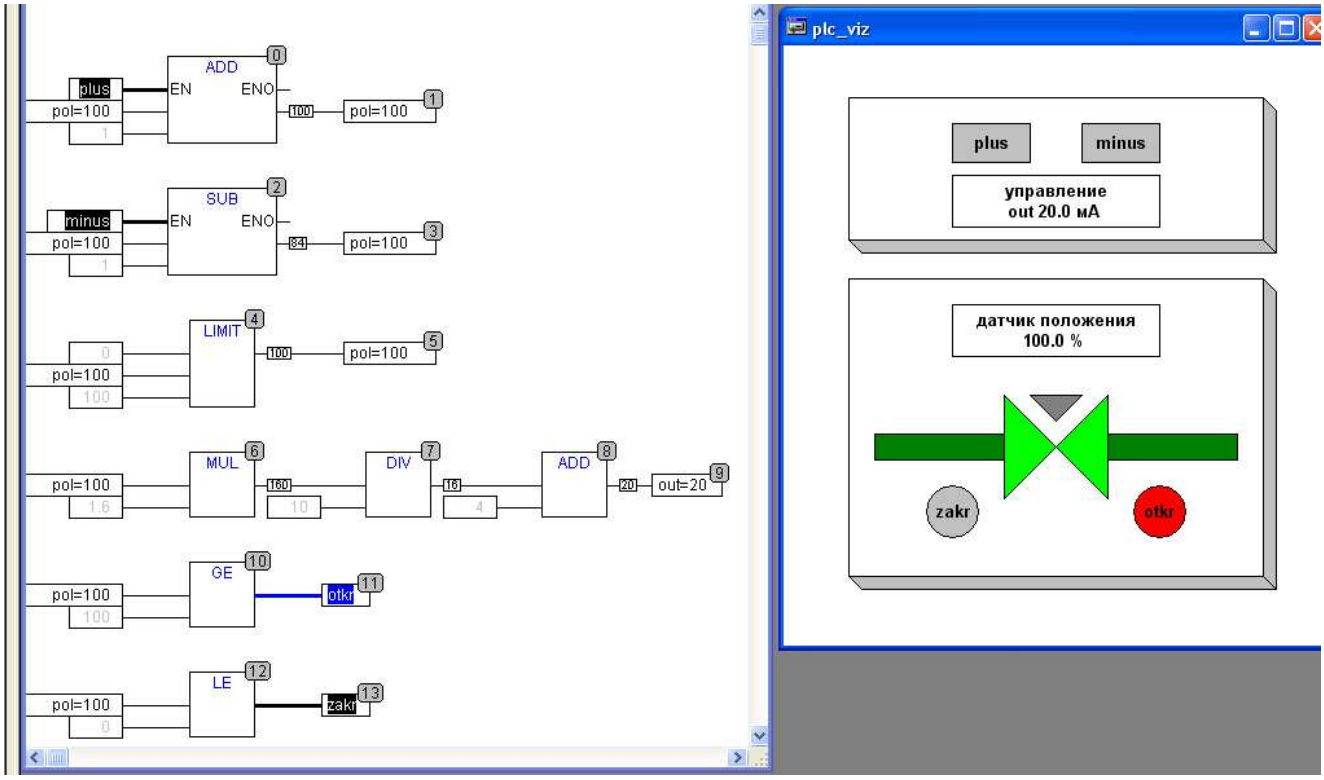

Рисунок 29 – Клапан відкритий повністю

Тест № 4

Закрити клапан, натискаючи кнопку minus (рисунок 30).

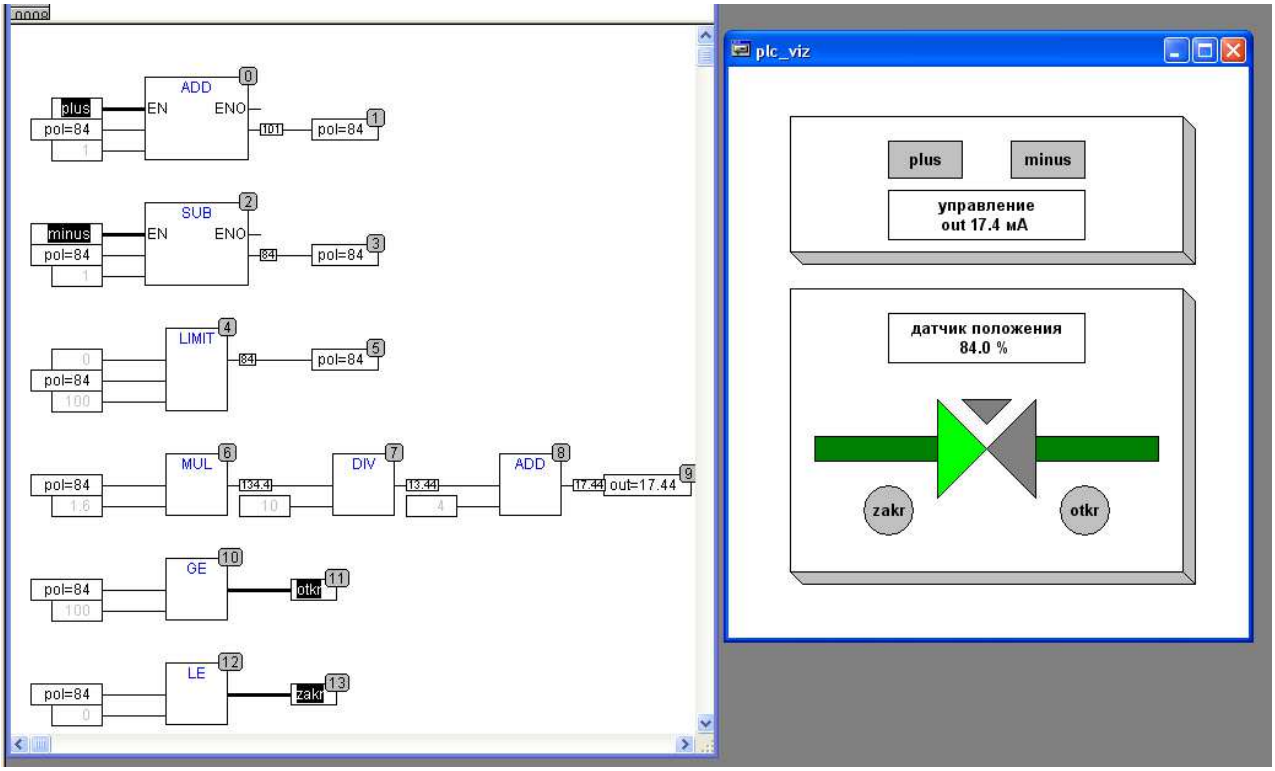

Рисунок 30 – Закриття клапану

Контрольні питання

1. Пояснить принцип роботи створеної системи керування.
2. Як виконувалось конфігурування ПЛК?
2. Які графічні елементи візуалізації були використані?
3. Як виконувалось конфігурування графічних елементів?

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Харазов В.Г. Интегрированные системы управления технологическими процессами. – СПб.: Профессия, 2009. – 592 с.

2. Денисенко В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. – М.: Горячая линия – Телеком, 2009. – 608 с.

3. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы программирования, М. :СОЛОН-ПРЕСС, 2008. - 256 с.

4. Деменков Н.П. Языки программирования промышленных контроллеров: учеб. пособ. / Деменков Н.П.; под ред. К.А. Пупкова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 172 с: ил.

5. Минаев И.Г., Самойленко В.В. Программируемые логические контроллеры: практическое руководство для начинающего инженера. – Ставрополь: АРГУС, 2009. – 100 с.

6. Минаев И. Г. Программируемые логические контроллеры в автоматизированных системах управления / И. Г. Минаев, В. М. Шарапов, В. В. Самойленко, Д. Г. Ушкур. 2-е изд., перераб. и доп. — Ставрополь: АГРУС, 2010. — 128 с.

7. Веб-сторінка фірми Smart Software Solutions GmbH виробника середовища CoDeSys. http://www.3s-software.com/.

8. Сайт виробника програмно-технічних засобів автоматизації – компанії ОВЕН: http://www.owen.ru, http://www-owen.com.ua.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
«МІКРОКОНТРОЛЕРИ ТА ЇХ ПРОГРАМУВАННЯ»
(для здобувачів вищої освіти денної та заочної форм навчання
спеціальності 151 Автоматизація та комп'ютерно–інтегровані технології)
Укладачі:
ПРОКАЗА Олена Іванівна
КУЗНЕЦОВА Олена Володимирівна
Оригінал - макет О.В. Кузнецова

Підписано до друку _____
Формат 60×84 1/16 . Папір типограф. Гарнітура Times.
Друк офсетний. Умовн. друк. арк ._____. Облік.-видавн.арк._____.
Тираж ____ екз. Вид. №_____. Замовл. №_____. Ціна договірна.

Видавництво Східноукраїнського національного університету імені
імені Володимира Даля

Адреса видавництва: м. Сєвєродонецьк, просп. Центральний, 59а
Телефон: +38 (050) 218 04 78, факс (064 52) 4 03 42

E-mail:vidavnictvosnu.ua@gmail.com