Навчально-методична література

Міністерство освіти і науки України
Тернопільський національний технічний університет
ім. Івана Пулюя
Кафедра комп'ютерно-інтегрованих технологій

Методичні вказівки до виконання практичних робіт

з дисципліни

«Автоматизовані системи керування технологічними процесами»

напряму підготовки 6.050202 «Автоматизація та комп'ютерно-інтегровані технології»

Тернопіль – 2016

УДК
621.36
К 89

Укладач:
Карташов В.В., канд. техн. наук, старший викладач.

Рецензенти:
П.Д. Стухляк, докт. техн. наук., професор,
Р.З. Золотий, канд. техн. наук, доцент.

Методичні вказівки розглянуто й затверджено на засіданні
методичного семінару кафедри комп'ютерно-інтегрованих технологій
Тернопільського національного технічного університету імені Івана Пулюя
протокол №1 від 29 серпня 2016 р.

Схвалено та рекомендовано до друку науково-методичною комісією
факультету прикладних інформаційних технологій та електроінженерії
Тернопільського національного технічного університету імені Івана Пулюя
протокол № 1 від 29 серпня 2016 р.

Методичні вказівки до виконання практичних робіт з дисципліни «Автоматизовані системи керування технологічними процесами» / Укладач : Карташов В.В. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 66 с.

УДК 621.36
К 89

Посібник складено з урахуванням матеріалів літературних джерел, наведених у переліку.

Відповідальний за випуск: *В.В. Карташов*, канд. техн. наук, ст. викл.

ЗМІСТ

ВСТУП

Метою проведення практичних робіт є ознайомлення студентів із будовою програмованих логічних контролерів (ПЛК), отримання навичок їх підключення та програмування. В процесі виконання лабораторних робіт студенти вивчають усі 6 (ST, IL, LD, FBD, SFC, CFC) мов програмування стандарту МЕК 61131-3 на основі програмного середовища CoDeSys V2.3 та мікроконтролерів ОВЕН.

CoDeSys - універсальний комплекс розробки програм для ПЛК. Його назва розшифровується як Controller Development System, він випускається німецькою фірмою 3S-Smart Software Solutions GmbH. Перша версія CoDeSys вийшла в 1994 році, а сьогодні доступна третя версія пакету, в якій підтримується об'єктно-орієнтоване програмування. До складу комплексу програмування ПЛК входять дві обов'язкові частини: середовище розробки програм і система виконання.

Середовище розробки програм CoDeSys функціонує на персональному комп'ютері (ПК) і не має конкретної апаратної прив'язки, тому з її допомогою можна створювати програми для будь-яких контролерів. Система виконання (CoDeSys SP) функціонує в контролері і встановлюється його виробником. Вона забезпечить завантаження коду програми, його виконання, а також налагодження функції. Середовище розробки програм CoDeSys є безкоштовним, і його можна завантажити з сайту виробника. Компанія 3S ліцензує тільки системи виконання. Існує русифікована версія CoDeSys. Цей програмний пакет може бути встановлений на комп'ютер, що працює під управлінням операційної системи Windows. Для завантаження написаної програми в будь-який конкретний ПЛК необхідний target-файл (файл цільової платформи), зазвичай поставляється виробником контролера. У цьому файлі знаходиться інформація про ресурси даного контролера, в тому числі – про його входи та виходи, типи і розташування даних в пам'яті і т.д. Для інсталяції target-файлів служить програма InstallTarget, яка встановлюється на комп'ютер разом з пакетом CoDeSys. CoDeSys застосовується для програмування багатьма фірмами, в Зокрема, WAGO, ABB, Beckhoff, і т.д.

Однією із переваг CoDeSys є наявність в ньому режиму емуляції, що дозволяє налагоджувати програми безпосередньо на комп'ютері, та не завантажуючи їх в контролер і без задіяння працюючого технологічного обладнання на стадії налагодження. Ця властивість дозволяє також широко використовувати CoDeSys в навчальному процесі.

Технічні рішення на базі CoDeSys і мікроконтролерів застосовуються, наприклад, в Німеччині для автоматизації вітряних енергетичних турбін Enercon з використанням контролерів Modular PLC. В Україні технічні рішення на базі CoDeSys і устаткування фірми Овен застосовуються при автоматизації технологічних процесів в сфері ЖКГ, в енергетиці, металургії, хімічній, харчовій та інших галузях промисловості.

В даний час існує некомерційна організація CoDeSys Automation Alliance (CAA) - об'єднання компаній- виробників ПЛК, що підтримують CoDeSys, в яке входять понад сімдесят фірм.

ПРАКТИЧНА РОБОТА №1

Тема: **“**Вивчення середовища CodeSys V2.3 та мови ST для програмування ПЛК ОВЕН 110-60М. Керування логічними входами та виходами”

Мета: Вивчити методи керування логічними входами та виходами контролера ОВЕН 110-60М.

Ціль роботи:

1. Знайомство з основними відомостями про мову програмування ST.
2. Знайомство з програмним середовищем CodeSyS V2.3.
3. Знайомство із можливостями режиму емуляції CodeSyS V2.3.

Короткі теоретичні відомості про мову ST

В даний час для програмування контролерів широко використовуються технологічні мови програмування стандарту МЕК 61131-3 (по-англійськи - IEC 61131-3, IEC - International Electrotechnical Commission, Міжнародна електротехнічна комісія): ST, IL, LD, FBD, SFC або їх модифікації. У CoDeSys використовуються шість мов - вищевказані стандартні мови та CFC, нестандартну мову, споріднену FBD.

Серед розглянутих мов дві - текстові (ST і IL), решта - графічні. Нижче дається їх коротка характеристика.

Мова ST (Structured Text, структурований текст) – текстова мова високого рівня, близька за структурою і синтаксисом до мови Pascal. На ST зручно записувати функції і функціональні блоки, описувати дії з аналоговими сигналами і числами з плаваючою крапкою, а також дії і переходи мови SFC. ST зручна для написання великих програм. Приклад простої програми на мові ST представлений на рис. 1.1 Дана програма може додавати два цілих числа, після чого за допомогою команди розгалуження (IF-THEN-ELSE) присвоює змінній деяке значення в залежності від значення отриманої суми.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     A: INT := 1;
0004     B: INT := 1;
0005     C: INT := 1;
0006     D: INT:=10;
0007 END_VAR
0001 A:=B+C;
0002 IF A>0 THEN
0003  D:=10;
0004 ELSE
0005  D:=0;
0006 END_IF;
```

Рис. 1.1 Приклад програми на мові ST.

ST являє собою набір інструкцій високого рівня, які можуть використовуватися в умовних операторах ("IF ... THEN ... ELSE") і в циклах (WHILE ... DO).

Вирази

Вираз - це конструкція, яка повертає певне значення після його обчислення. Вираз складається з операторів і операндів. Операндом може бути константа, змінна, функціональний блок або інший вираз.

Обчислення виразів

Обчислення виразів виконується згідно з правилами пріоритету. Оператор з найвищим пріоритетом виконується першим, оператор з більш низьким пріоритетом - другим і т.д., поки не будуть виконані всі оператори.

Оператори з однаковим пріоритетом виконуються зліва направо.

У наступній таблиці наведено список ST операторів, розташованих в порядку пріоритету.

Операція	**Позначення**	**Пріоритет**
Вираз в дужках	(вираз)	Найвищий.
Виклик функції	Ім'я функції (Список параметрів)	
Піднесення до степеня	EXPT	
Числове доповнення	NOT	
множення	*	
ділення	/	
залишок від ділення	MOD	
додавання	+	
віднімання	-	
порівняння	<,>, <=,> =	
нерівність	<>	
рівність	=	
логічне І	AND	
логічне виключає АБО	XOR	
Логічне АБО	OR	Найнижчий

Оператор присвоювання.

Перед оператором присвоювання знаходиться операнд (змінна або адреса), якій присвоюється значення виразу, що стоїть після оператора присвоювання.

Приклад:

Var1: = Var2 * 10;

Після виконання цієї операції Var1 приймає значення в десять разів більше, ніж Var2.

Завдання для розробки.

Розробити керуючу програму для керування логічними входами та виходами контролера ОВЕН. При активізації вибраного входу (наприклад DI6) повинна активізуватись така кількість виходів, який номер відповідає активованому. У візуалізації передбачити активацію входів черех кнопки, та активацію виходів через лампи (табл.1).

Таблиця 1. Варіанти завдань

№ п/п	Входи, які підлягають активації	Активні виходи при активному вході відповідно		
1	DI1 DI3 DI5	DO1	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5
2	DI3 DI6DI4	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5 DO6	DO1 DO2 DO3 DO4
3	DI2 DI7 DI9	DO1 DO2	DO1 DO2 DO3 DO4 DO5 DO6 DO7	DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 DO9
4	DI2 DI4 DI1	DO1 DO2	DO1 DO2 DO3 DO4	DO1
5	DI3 DI6 DI4	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5 DO6	DO1 DO2 DO3 DO4
6	DI2 DI6 DI8	DO1 DO2	DO1 DO2 DO3 DO4 DO5 DO6	DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8
7	DI1 DI3 DI6	DO1	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5 DO6
8	DI7 DI4 DI9	DO1 DO2 DO3 DO4 DO5 DO6 DO7	DO1 DO2 DO3 DO4	DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 DO9

Порядок виконання.

Виконаємо для прикладу 1-й варіант завдання. Створіть новий проект обравши пункт верхнього меню **File / New**. Оберіть модель мікроконтролера **ПЛК Овен 110_60М (**рис. 1.2**).**

Рисунок 1.2

Натисніть **ОК**. При створенні кожного нового проекту необхідно обирати модель контролера. Якщо цього не зробити, бібліотеки із стандартним набором операторів (таймери, тригери, лічильники і т.д.) будуть не доступні, і їх доведеться підключати вручну.

Не змінюйте назву нового проекту **PLC_PRG**. Обираємо мову програмування для нього **ST**, тип програмного компоненту **Program (**рис. 1.3**).**

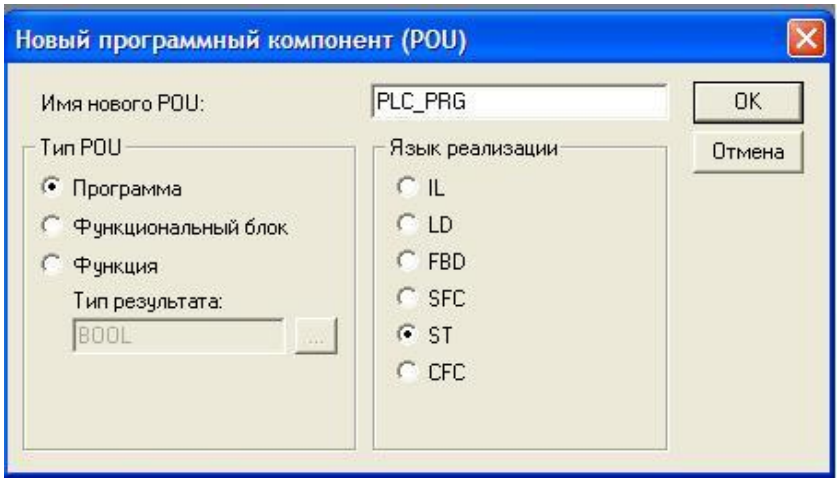

Рисунок 1.3

Згідно варіанту нашого завдання нам потрібно щоб кнопка **DI1** вмикала лампу **DO1**. Використаємо умову «**Якщо - тоді**» (**IF - THEN**). Оголосіть 2 змінних типу **BOOL** з назвою **DI1** та **DO1** набравши їх на клавіатурі у вікні оголошення змінних між позначками **VAR** та **END_VAR (**рис. 1.4**).** Наберіть у вікні програми програмний код зображений на рис. 1.5:

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     DI1: BOOL;
0004     DO1: BOOL;
0005 END_VAR
```

Рисунок 1.4

```
0001 IF DI1=TRUE THEN
0002 DO1:=TRUE;
0003 END_IF
```

Рисунок 1.5

Перевіримо цю частину програмного коду у візуалізації. Перейдіть на третю закладку лівого вікна CoDeSys (Візуалізації). Добавте об’єкт візуалізації із контекстного меню (**Add Object**). Введіть ім’я об’єкту (наприклад **Visual**). З’явиться вікно графічного редактора. Побудуйте кнопку з допомогою команди **Вставка / Кнопка (Insert / Button)**. Для цього у вікні редактора візуалізації натисніть ЛКМ і розтягніть прямокутник до потрібної висоти та ширини, після чого відпустіть ЛКМ. Далі побудуйте лампочку. Оберіть пункт **Вставка / Еліпс (Insert / Ellipse).** Для цього у вікні редактора візуалізації натисніть ЛКМ і розтягніть еліпс до потрібного розміру, після чого відпустіть ЛКМ (рис. 1.6).

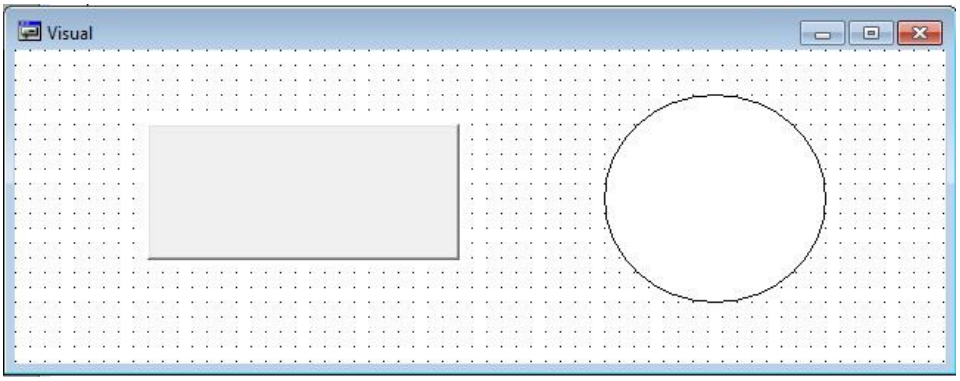

Рисунок 1.6

Ми побудували графічні об’єкти. Тепер з них треба зробити кнопку та лампочку. Почнемо із кнопки. Двічі клікніть ЛКМ на прямокутнику. Відкриється діалогове вікно налаштування елементу візуалізації. Задайте у вікні конфігурації елемента в категорії **Текст** слово **DI1**. Це слово відображатиметься на нашому графічному об’єкті як надпис. Збільшіть розмір шрифту натиснувши **Шрифт (Font…) (**рис. 1.7**)**

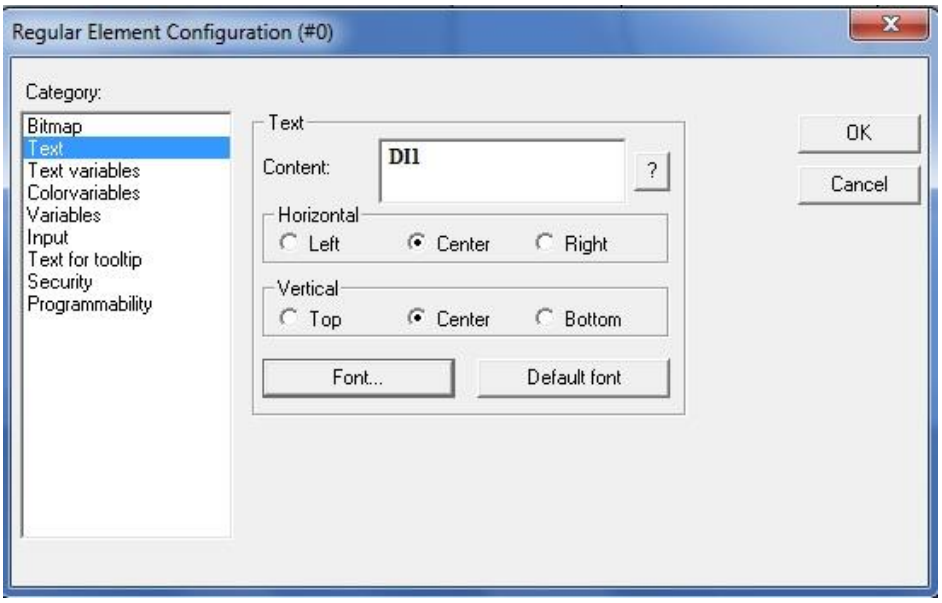

Рисунок 1.7

Щоб змінна **DI1** змінювала своє значення при натисканні на нашу кнопку нам треба в категорії вводу (**Input**) поставити галочку **Змінна-переключення** (**Toggle variable**) і поставити курсор в полі справа. Введіть назву нашого програмного компоненту **PLC_PRG** і поставте крапку. CoDeSys запропонує вибрати вже оголошені на цей момент змінні із цього програмного компоненту. У випавшому списку виберіть подвійном кліком ЛКМ змінну **DI1** (рис. 1.8).

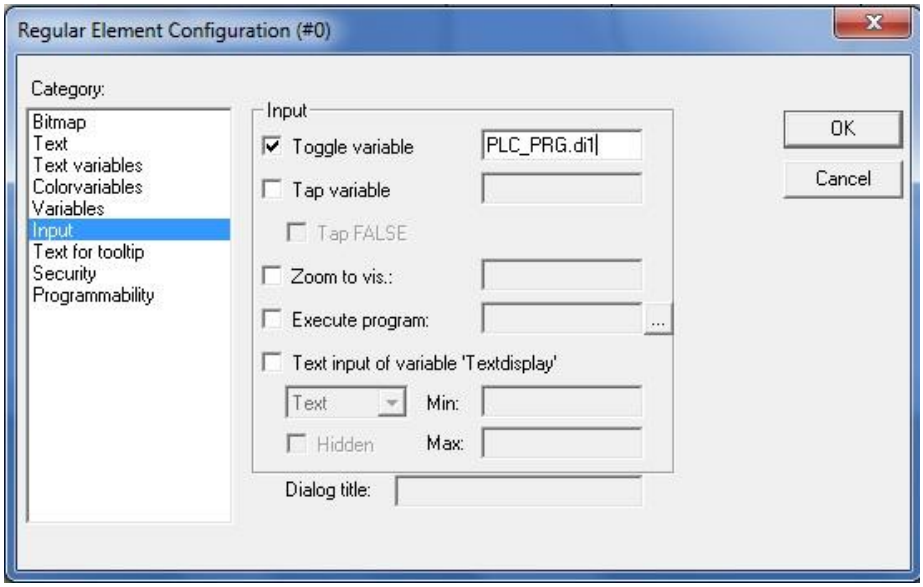

Рисунок 1.8

На цьому вікно конфігурації елемента «Кнопка» завершено. Закрийте вікно натиснувши **ОК**.

Далі зробимо із нашого круга лампочку. Лампочка в нашому випадку - це графічний об’єкт, що змінює колір із тьмяного на яскравий. Двічі клікніть ЛКМ на крузі. Відкриється діалогове вікно налаштування елементу візуалізації. Задайте у вікні конфігурації елемента в категорії **Текст** слово **DO1** та збільшіть розмір шрифту як ми це робили раніше. Перейдіть в категорію **Кольори** (**Colors**). Задайте будь-який тьмяний колір заливки елементу (**Inside**), наприклад сірий. Для «збудженого» стану необхідно обрати інший колір (**Alarm color**), якийсь яскравий, наприклад жовтий. Тепер наша лампочка буде сірого кольору у вимкненому стані, і жовтого у ввімкнутому. Далі треба зробити щоб ці кольори змінювалися коли змінна **DO1** змінюватиме своє значення. Для цього перейдіть в категорію змінних (**Variables**). Клікніть ЛКМ в полі зміни кольору (**Change color**) і скористуйтесь асистентом вводу натиснувши «**F2**» (рис. 1.9).

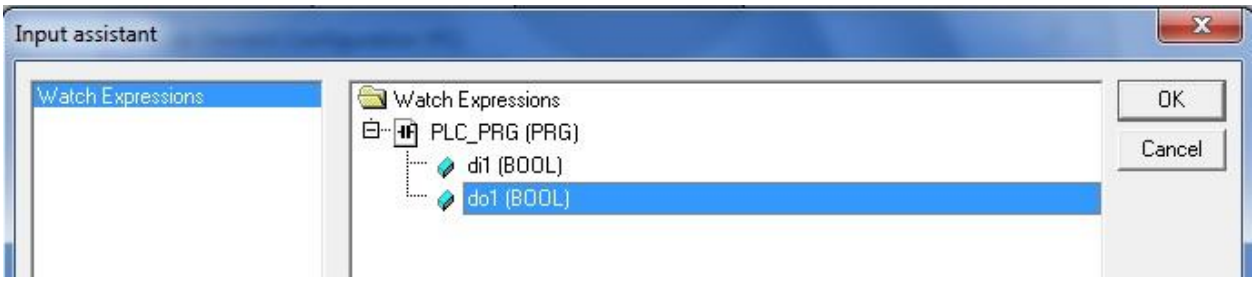

Рисунок 1.9

Виберіть подвійним кліком ЛКМ змінну **DO1** із списку. Тепер наш графічний об’єкт змінюватиме колір при зміні значення змінної **DO1.** Тепер вікно конфігурування елементу можна закривати. Для цього натисніть **ОК**. В результаті наш круг буде відображатися сірим кольором при значенні змінної **DO1** FALSE, і жовтим при значенні TRUE.

Тепер наше вікно графічного редактора має такий вигляд (рис. 1.10):

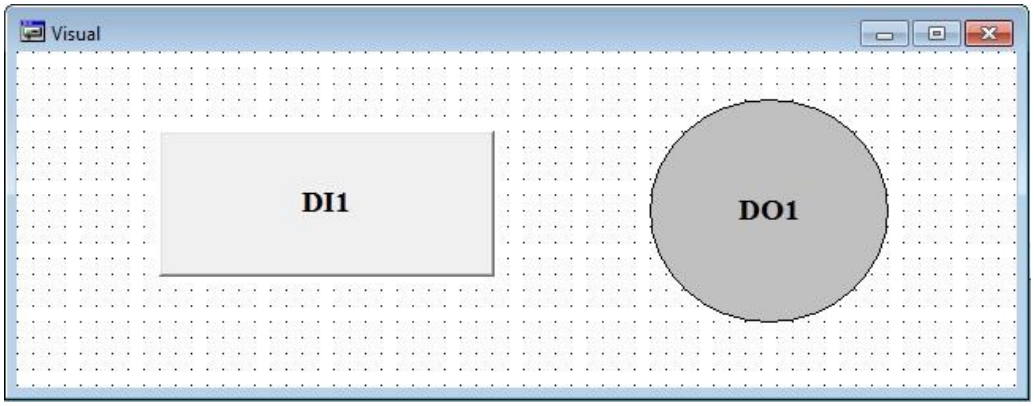

Рисунок 1.10

Перевіримо наш проект на помилки. Відкомпілюйте проект повністю командою верхнього меню «**Проект / Компілювати все**» (**Project / Rebuild all**), або скористайтеся клавішею «**F11**». Якщо ви все виконали вірно, то в нижній частині вікна повідомлень має з’явитись надпис «0 errors» (рис. 1.11)

Рисунок 1.11

В іншому випадку необхідно виправити допущені помилки. В цьому допоможуть повідомлення про помилки.

Спробуємо, що у нас вийшло. Для цього служить режим емуляції. Поставте галочку у верхньому меню в пункті **Онлайн / Режим емуляції (Online / Simulation mode).** Тепер оберіть пункт **Онлайн / Підключення, (Online / Login),** потім **Онлайн / Старт (Online / Run)**. Натисніть на кнопку **DI1,** при цьому лампочка **DO1** має змінити свій колір на жовтий. Знову натисніть на кнопку **DI1,** при цьому лампочка **DO1** має змінити свій колір знову на сірий. Як бачимо цього не відбувається. Це тому, що ми вказали коли змінна **DO1** має набути значення TRUE, але не вказали коли змінна **DO1** має вернутися у FALSE.

Відключіться від «віртуального ПЛК» обравши пункт верхнього меню **Онлайн / Відключення, (Online / Logout).** Поверніться до програмного компоненту **PLC_PRG**.

Допишіть програмний код наступним чином (рис. 1.12). Поверніться у візуалізацію та знову запустіть проект. Перед цим не забудьте перевірити проект на помилки (скомпілювати). Що ж, тепер при натисненні на кнопку лампочка загорається, а при відтисненні гасне. Чудово! Можемо приступати до програмування інших кнопок. Поверніться до програмного компоненту **PLC_PRG**.

Спробуємо аналогічно запрограмувати наступну кнопку **DI3**. Доповніть програмний код наступним чином (рис. 1.13):

```
0001 IF DI1=TRUE THEN
0002 DO1:=TRUE;
0003 ELSE
0004 DO1:=FALSE;
0005 END_IF
```

Рисунок 1.12

```
0001 IF DI1=TRUE THEN
0002 DO1:=TRUE;
0003 ELSE
0004 DO1:=FALSE;
0005 END_IF
0006
0007 IF DI3=TRUE THEN
0008 DO1:=TRUE;
0009 DO2:=TRUE;
0010 DO3:=TRUE;
0011 ELSE
0012 DO2:=FALSE;
0013 DO3:=FALSE;
0014 END_IF
```

Рисунок 1.13

Скомпілюйте проект, перейдіть у візуалізацію та знову запустіть проект. Спробуйте як працюють кнопки.

Запрограмуйте третю кнопку **DI5** самостійно.

Додаткові завдання.

1. Запрограмуйте алгоритм роботи кнопок згідно вашого варіанту використовуючи **логічне АБО (OR)**. Приклад для двох кнопок наведено нижче (рис. 1.14):

```
0001 IF DI1=TRUE OR DI3=TRUE THEN
0002 DO1:=TRUE;
0003 ELSE
0004 DO1:=FALSE;
0005 END_IF
0006
0007 IF DI3=TRUE THEN
0008 DO2:=TRUE;
0009 DO3:=TRUE;
0010 ELSE
0011 DO2:=FALSE;
0012 DO3:=FALSE;
0013 END_IF
```

Рисунок 1.14

2. Запрограмуйте алгоритм роботи кнопок згідно вашого варіанту використовуючи умовний оператор **ELSIF**. Приклад для двох кнопок наведено нижче (рис. 1.15):

```
0001 IF DI1=TRUE THEN
0002 DO1:=TRUE;
0003 DO2:=FALSE;
0004 DO3:=FALSE;
0005 ELSIF DI3=FALSE THEN
0006 DO1:=FALSE;
0007 END_IF
0008
0009 IF DI3=TRUE THEN
0010 DO1:=TRUE;
0011 DO2:=TRUE;
0012 DO3:=TRUE;
0013 ELSIF DI1=FALSE THEN
0014 DO2:=FALSE;
0015 DO3:=FALSE;
0016 END_IF
```

Рисунок 1.15

ПРАКТИЧНА РОБОТА №2

Тема: “Вивчення середовища CodeSys V2.3 та мови LD для програмування ПЛК ОВЕН 110-60М. Керування логічними входами та виходами”

Мета: Вивчити методи керування логічними входами та виходами при програмуванні контролера ОВЕН 110-60М.

Ціль роботи:

1. Знайомство з основними відомостями про мову програмування LD.
2. Знайомство з програмним середовищем CodeSyS V2.3.
3. Знайомство із можливостями режиму емуляції CodeSyS V2.3.

Короткі теоретичні відомості про мову LD

Мова LD (Ladder Diagram, мова релейних діаграм або релейно-контактних схем) призначена для реалізації програм логічного управління, при цьому в даній мові використовуються тільки логічні змінні (тобто, змінні типу Bool). Основними елементами програми на даній мові є контакти і обмотки. Приклад програми, що реалізує логічний елемент «І» на три входи, показаний на рис. 2.1.

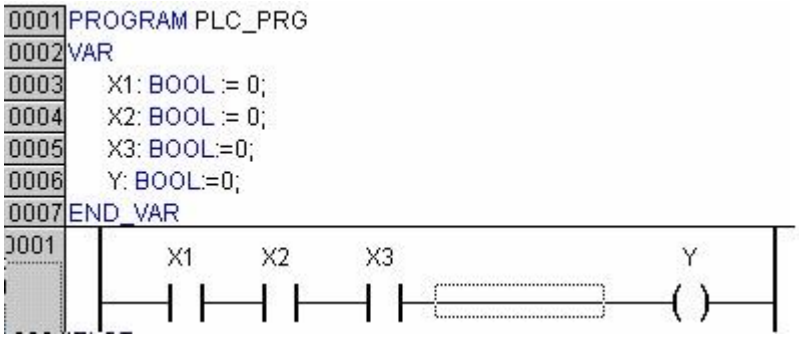

Рис. 2.1 Приклад програми на мові LD.

Мова релейних або релейно-контактних схем (РКС) - графічна мова, яка реалізує структури електричних ланцюгів. Найкраще LD підходить для побудови логічних перемикачів, але досить легко можна створювати і складні ланцюги - як в FBD. Крім того, LD досить зручна для управління іншими компонентами POU. Діаграма LD складається з ряду ланцюгів. Ліворуч і праворуч схема обмежена вертикальними лініями - шинами живлення. Між ними розташовані ланцюги, 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 «БРЕХНЯ». Кожному контакту відповідає логічна змінна. Якщо змінна має значення «ІСТИНА», то стан передається через контакт. Інакше праве з'єднання отримує значення вимкнено ("OFF").

Контакт

Контакти позначаються двома паралельними лініями і можуть мати стан "ON" або "OFF". Ці стани відповідають значенням ІСТИНА або БРЕХНЯ. 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 інвертованим. Такий контакт позначається за допомогою символу | / | і передає стан "ON", якщо значення змінної БРЕХНЯ.

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 якраз і відображається на обмотці і відповідно на логічній змінній обмотки (ІСТИНА / БРЕХНЯ). Обмотки також можуть бути інверсними. 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 обмотки

Обмотки можуть бути із «самофіксацією» типів SET і RESET. Обмотки типу SET позначаються буквою "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 БРЕХНЯ, то вона назавжди (до встановлення S) зберігає його.

Завдання для розробки.

Розробити керуючу програму для керування логічними входами та виходами контролера ОВЕН. При активізації вибраного входу (наприклад DI6) повинна активізуватись така кількість виходів, який номер відповідає активованому. У візуалізації передбачити активацію входів черех кнопки, та активацію виходів через лампи (табл. 2).

Таблиця 2. Варіанти завдань

№ п/п	Входи, які підлягають активації	Активні виходи при активному вході відповідно		
1	DI1 DI3 DI5	DO1	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5
2	DI3 DI6DI4	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5 DO6	DO1 DO2 DO3 DO4
3	DI2 DI7 DI9	DO1 DO2	DO1 DO2 DO3 DO4 DO5 DO6 DO7	DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 DO9
4	DI2 DI4 DI1	DO1 DO2	DO1 DO2 DO3 DO4	DO1
5	DI3 DI6 DI4	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5 DO6	DO1 DO2 DO3 DO4
6	DI2 DI6 DI8	DO1 DO2	DO1 DO2 DO3 DO4 DO5 DO6	DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8
7	DI1 DI3 DI6	DO1	DO1 DO2 DO3	DO1 DO2 DO3 DO4 DO5 DO6
8	DI7 DI4 DI9	DO1 DO2 DO3 DO4 DO5 DO6 DO7	DO1 DO2 DO3 DO4	DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 DO9

Порядок виконання.

Виконаємо для прикладу 1-й варіант завдання. Створіть новий проект обравши пункт верхнього меню **File / New**. Оберіть модель мікроконтролера **ПЛК Овен 110_60М**. Натисніть **ОК**. Не змінюйте назву нового проекту **PLC_PRG**. Обираємо мову програмування для нього **LD**, тип програмного компоненту **Program** (рис. 2.2).

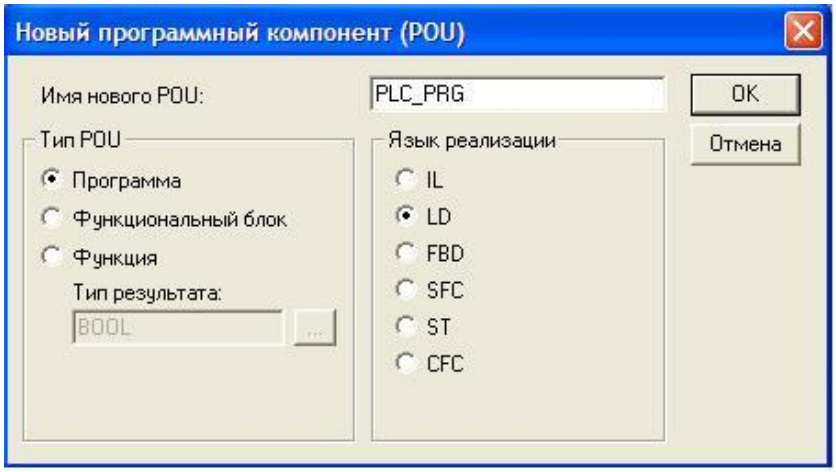

Рисунок 2.2

Згідно варіанту нашого завдання нам потрібно щоб кнопка **DI1** вмикала лампу **DO1**. Виділіть рядок для програмування на мові LD клікнувши на ньому

ЛКМ. Справа на рядку з'явиться пунктирна рамка. Це означає, що рядок активний, тобто контакти і котушки з панелі інструментів розташовуватимуться на ньому. Клікніть ЛКМ на контакті (**Contact**) на панелі інструментів, або із верхнього меню (**Insert / Contact**). Контакт автоматично розташується в лівій частині рядка. Замініть **???** на ім'я першої змінної **DI1.** Натисніть **Enter**. У діалоговому вікні оголошення змінних вкажіть клас **VAR** і тип **BOOL** (рис. 2.3)

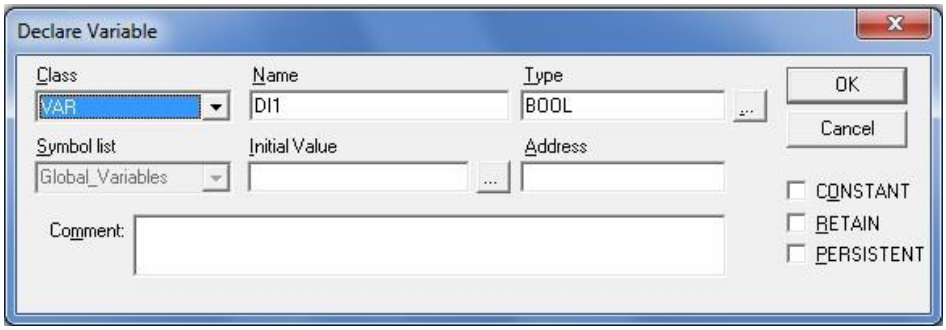

Рисунок 2.3

Змінна DI1 автоматично з'явиться у верхньому вікні змінних.

Вставте котушку (**Insert / Coil**). Котушка автоматично розташується в правій частині рядка. Замініть **???** на ім'я змінної **DO1**. Натисніть **Enter**. У діалоговому вікні оголошення змінних вкажіть клас **VAR** і тип **BOOL.**

Тепер наша програма виглядає наступним чином (рис. 2.4):

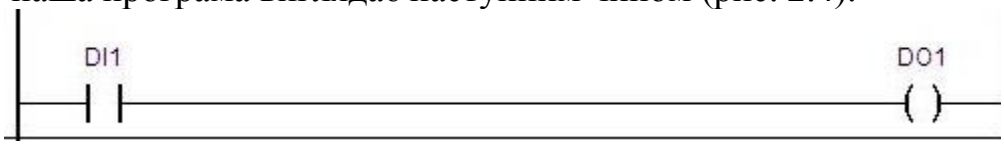

Рисунок 2.4

Перевіримо цю частину програмного коду у візуалізації. Перейдіть на третю закладку лівого вікна CoDeSys (Візуалізації). Добавте об'єкт візуалізації із контекстного меню (**Add Object**). Введіть ім'я об'єкту (наприклад **Visual**). З'явиться вікно графічного редактора. Побудуйте кнопку та лампочку і задайте їхню конфігурацію як ми це робили у ПР№1.

Тепер наше вікно графічного редактора має такий вигляд (рис. 2.5):

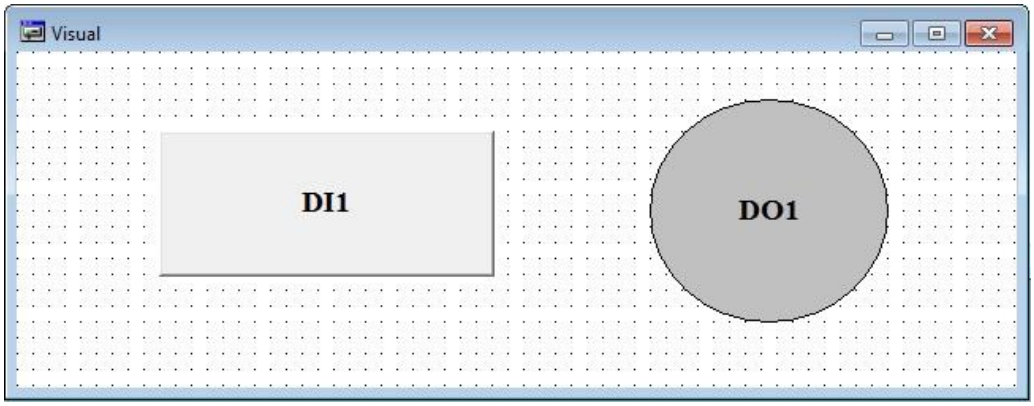

Рисунок 2.5

Перевіримо наш проект на помилки. Відкомпілюйте проект повністю командою верхнього меню «**Проект / Компілювати все**»(**Project / Rebuild all**),

або скористайтеся клавішею «**F11**». Якщо ви все виконали вірно, то в нижній частині вікна повідомлень має з'явитись надпис «0 errors» (рис. 2.6)

Рисунок 2.6

В іншому випадку необхідно виправити допущені помилки. В цьому допоможуть повідомлення про помилки.

Спробуємо, що у нас вийшло. Для цього служить режим емуляції. Поставте галочку у верхньому меню в пункті **Онлайн / Режим емуляції (Online / Simulation mode).** Тепер оберіть пункт **Онлайн / Підключення, (Online / Login),** потім **Онлайн / Старт (Online / Run)**. Натисніть на кнопку **DI1,** при цьому лампочка **DO1** має змінити свій колір на жовтий. Знову натисніть на кнопку **DI1,** при цьому лампочка **DO1** має змінити свій колір знову на сірий.

Тепер запрограмуємо інші кнопки. Відключіться від «віртуального ПЛК» обравши пункт верхнього меню **Онлайн / Відключення, (Online / Logout).** Поверніться до програмного компоненту **PLC_PRG**. Добавте другий рядок програми обравши пункт верхнього меню **Вставити / Ланцюжок після (Insert / Network After)**. Вставте в нього контакт **DI3.** Кнопка **DI3** має вмикати три лампочки - **DO1, DO2** та **DO3.** Тому вставте три котушки з відповідними назвами. Для всіх цих змінних оберіть клас **VAR** і тип **BOOL** (рис. 2.7)**.**

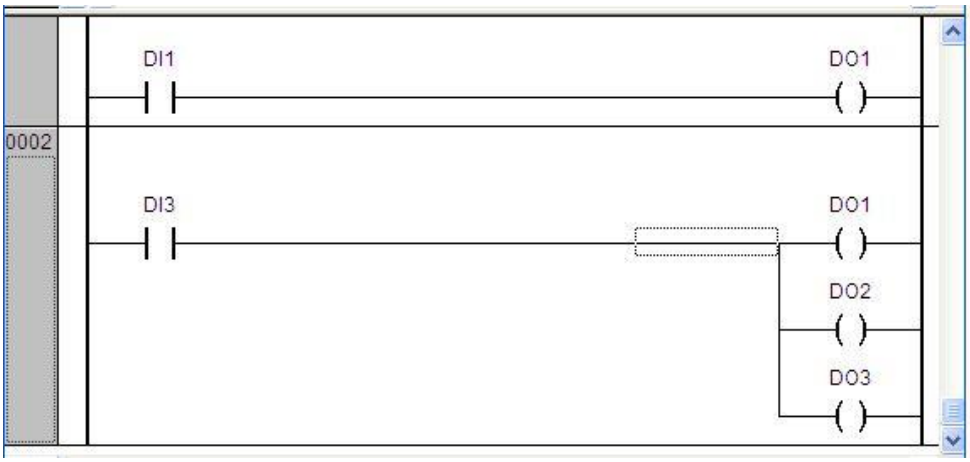

Рисунок 2.7

Поверніться у вікно редактора візуалізації. Скопіюйте створений круг командою **Edit / Copy** або з використанням комбінації гарячих клавіш **Ctrl + C**. Вставте скопійований круг один раз командою **Edit / Paste** (або **Ctrl + V)**. Відредагуйте конфігурацію нового круга: змініть текст «**DO1**» на «**DO2**», а в категорії змінних в полі зміни кольору з допомогою асистенту вводу «**F2**» змініть змінну **DO1** на змінну **DO2.** Аналогічним чином скопіюйте ще один круг для лампочки **DO3**. Скопіюйте та відредагуйте кнопку, змінивши текст та змінну в полі **Змінна-переключення** (**Toggle variable**) на **DI3** (рис. 2.8).

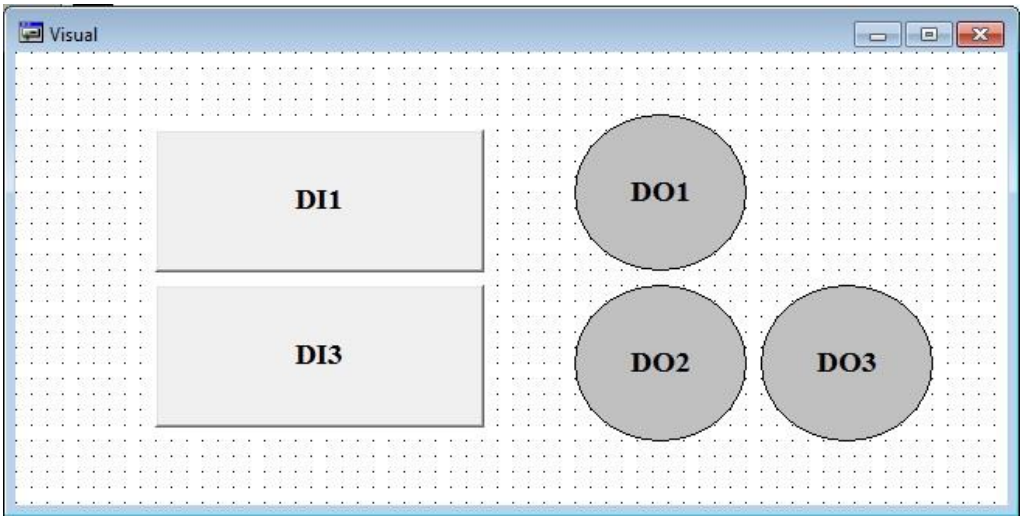

Рисунок 2.8

Перевірте проект на помилки та запустіть. Натисніть та відтисніть спочатку кнопку **DI1** а потім кнопку **DI3.** Як бачимо кнопка **DI3** засвічує всі три лампочки, як нам і потрібно, а кнопка **DI1** перестала працювати. Це відбувається тому, що ми пов'язали першу лампочку (**DO1**) і з першою (**DI1**) і з другою (**DI3**) кнопками. Відповідно, при натисканні першої кнопки **DI1** ця лампочка не загорається, бо не натиснута друга кнопка **DI3**. При натисканні другої кнопки **DI3** загораються всі три лампочки, в тому числі і перша. Це тому, що ця команда є останньою. Щоб робота однієї кнопки не втручалась в роботу іншої, нам слід розділити їх дію таким чином: першу лампочку **DO1** вмикає як кнопка **DI1** так і кнопка **DI3**. Тобто або змінна **DI1** або змінна **DI3** має подавати сигнал на котушку **DO1.** А котушки **DO2** і **DO3** вмикає лише змінна **DI3.**

Поверніться у вікно програмного компоненту **PLC_PRG** та змініть програму. Щоб видалити котушку слід виділити її клікнувши на ній ЛКМ та натиснути **Delete**. Щоб подати сигнал на котушку **DO1** через контакт **DI1** АБО контакт **DI3** скористайтесь командою **Вставити / Паралельний контакт (Insert / Parallel contact)**.

Таким чином наша програма тепер має виглядати так (рис. 2.9):

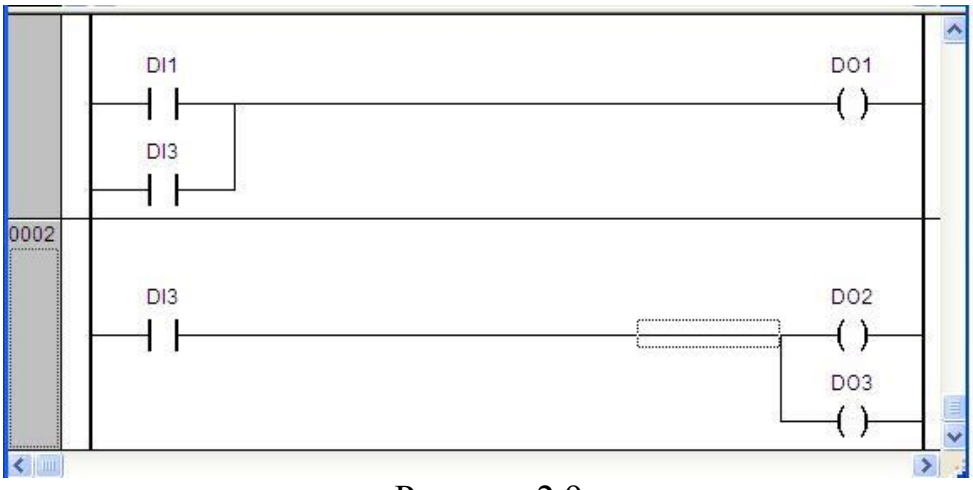

Рисунок 2.9

Перевірте проект на помилки виконавши компіляцію, поверніться у візуалізацію, запустіть проект на виконання, та переконайтеся, що тепер обидві

кнопки працюють – кнопка **DI1** вмикає лампочку **DO1**, а кнопка **DI3** вмикає лампочки **DO1, DO2** та **DO3**.

Поверніться у вікно програмного компоненту PLC_PRG та запрограмуйте третю кнопку **DI5** самостійно.

Додаткове завдання.

Запрограмуйте алгоритм роботи кнопок згідно вашого варіанту використовуючи котушки **SET** та **RESET**. Приклад для двох кнопок наведено нижче (рис. 2.10):

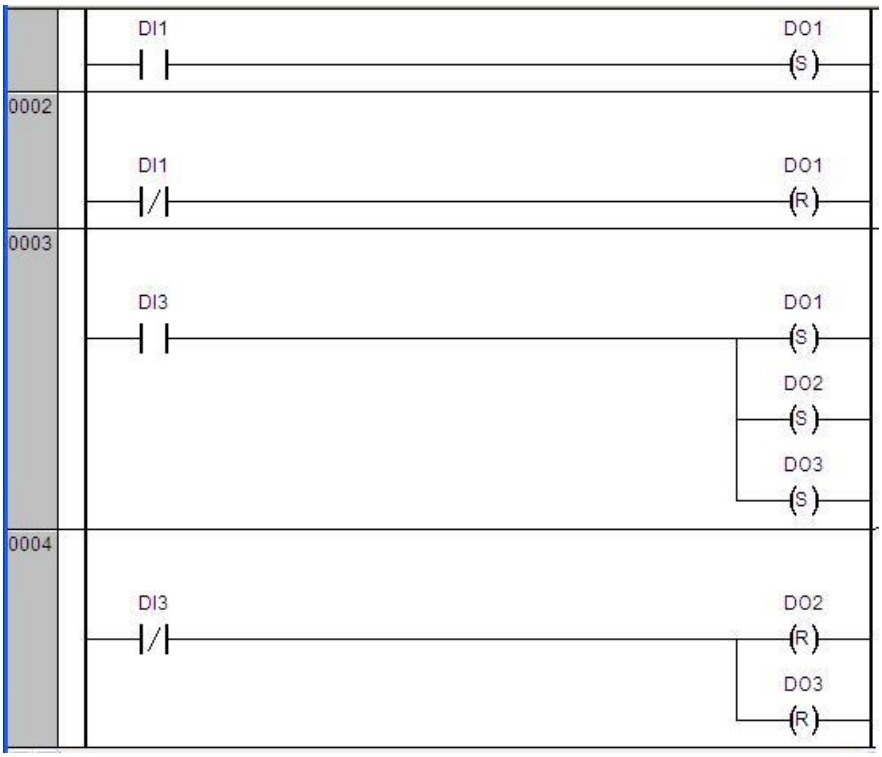

Рисунок 2.10

ПРАКТИЧНА РОБОТА №3

Тема: «Вивчення середовища CodeSys V2.3 та мови CFC для програмування ПЛК ОВЕН.»

Мета: Розробити програму для обчислення арифметичних функцій в середовищі CodeSys з використанням операторів порівняння.

Ціль роботи:

1. Знайомство з основними відомостями про мову програмування CFC.
2. Одержання навичок роботи з арифметичними операторами, операторами порівняння, операторами LIMIT, SEL та MUX.

Короткі теоретичні відомості про мову CFC

Мова CFC (Continuous Function Chart, мова безперервних функціональних схем) - графічна мова, споріднена до мови FBD. Ця мова не входить в стандарт МЕК 61131-3. У програмах, написаних на CFC, функціональні блоки розміщуються в області коду довільно, крім того, користувач може сам встановити порядок їх виконання. Приклад програми на мові CFC наведено на рис. 3.1. Дана програма реалізує вираз: Y=A/B+C*D

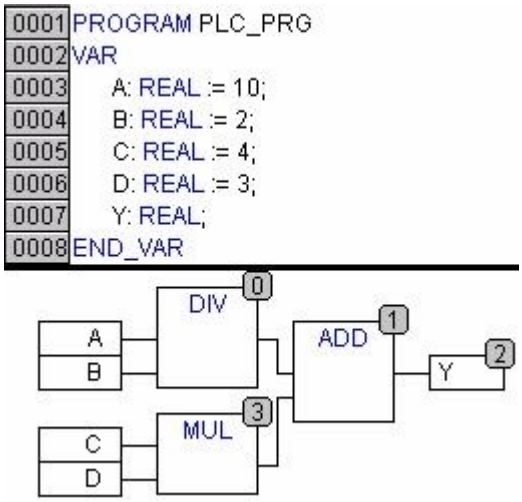

Рис. 3.1 Приклад програми на мові CFC.

Завдання.

Розробити програму використовуючи мову програмування CFC для обчислення арифметичних функцій з використанням операторів вибору та порівняння (табл. 3).

1-4. Програма повинна видавати значення змінної типу «Word» **w1** на виході оператора «MUX». Якщо значення змінної на виході оператора LIMIT більше 75, змінна типу «BOOL» на вході оператора «SEL» повинна встановлюватись у TRUE, і подавати на вихід оператора «SEL» значення 3 (якщо FALSE тоді 5), (для варіантів 1 - 4). Змінна на виході оператора LIMIT має знаходитись в межах від 0 до 100 (для 1 і

2 варіанту), і від 5 до 80 (для 3 і 4 варіанту). Це значення є мінімальним значенням між добутком змінних ((r1+r2+r3)/i1)*i2 та добутком ((r4-r5)/i3)*i4.

5-8. Програма повинна видавати значення змінної типу «Word» **w1** на виході оператора «MUX». Якщо значення змінної на виході оператора LIMIT менше 20, змінна типу «BOOL» на вході оператора «SEL» повинна встановлюватись у FALSE, і подавати на вихід оператора «SEL» значення 2 (якщо TRUE тоді 4), (для варіантів 5 - 8). Змінна на виході оператора LIMIT має знаходитись в межах від 5 до 25 (для 5 і 6 варіанту), і від 3 до 56 (для 7 і 8 варіанту). Це значення є максимальним значенням між добутком змінних ((r1+r2-r3)/i1+i3)*i2 та добутком ((r4-r5-r1)/(i3-i2))*i4.

Таблиця 3. Значення змінних.

№ вар.	Змінні типу REAL					Змінні типу INT			
	R1	R2	R3	R4	R5	I1	I2	I3	I4
1	8	2	1	12	10	9	5	3	11
2	7	1	2	15	12	8	4	3	12
3	8	3	1	16	11	7	6	2	13
4	7	2	3	17	10	9	5	1	14
5	8	1	2	18	9	7	4	3	15
6	7	3	1	15	12	8	6	2	16
7	8	1	3	12	11	9	5	1	17
8	7	2	2	13	10	7	4	3	18

Порядок виконання.

Виконаємо для прикладу 1-й варіант. Створіть новий проект обравши пункт верхнього меню **File / New**. Оберіть модель мікроконтролера **ПЛК Овен 110_60М**. Натисніть **ОК**. Не змінюйте назву нового проекту **PLC_PRG**. Обираємо мову програмування для нього **CFC**, тип програмного компоненту **Program** (рис. 3.2).

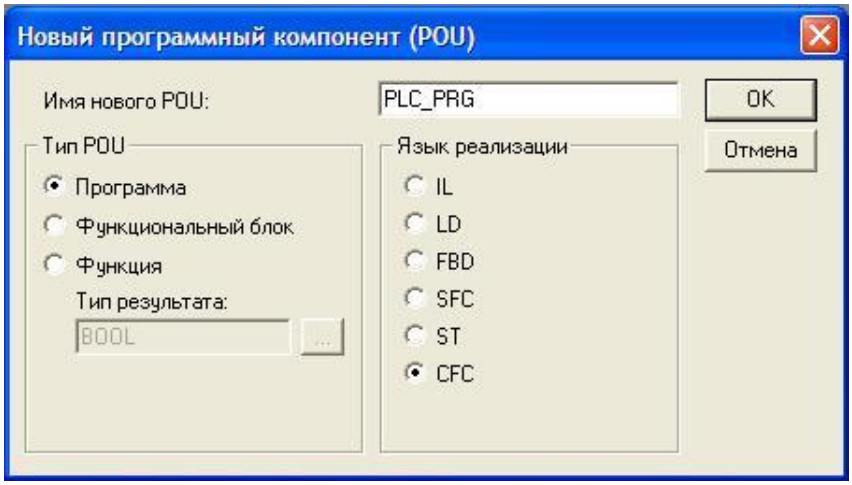

Рисунок 3.2

Спочатку нам потрібно обчислити результати виразів ((r1+r2+r3)/i1)*i2 та ((r4-r5)/i3)*i4.

Виконайте команду верхнього меню **Insert / Box (Вставити / Елемент)** або скористайтесь комбінацією гарячих клавіш **Ctrl + B**. По замовчуванні це буде логічне «і» (**AND**) (рис. 3.3). Клікніть ЛКМ на слові «**AND**» і змініть його на «**ADD**» (операція додавання) (рис. 3.4). Нам треба додати три числа, тому добавимо ще один вхід. Клікніть ПКМ на пунктирній рамці блоку «**ADD**» і виберіть в контекстному меню пункт **«Input of box» (Вхід блоку)** (рис. 3.5). Тепер розташуємо зліва блоку три входи (**Insert / Input**) або **Ctrl + I** (рис. 3.6)**.**

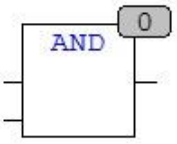

Рисунок 3.3

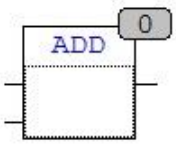

Рисунок 3.4

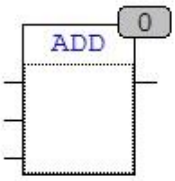

Рисунок 3.5

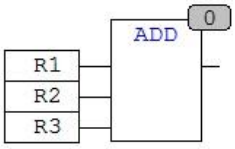

Рисунок 3.6

Замість знаків питання введіть змінні «R1», «R2» та «R3» та задайте їм тип **REAL** (рис. 3.7).

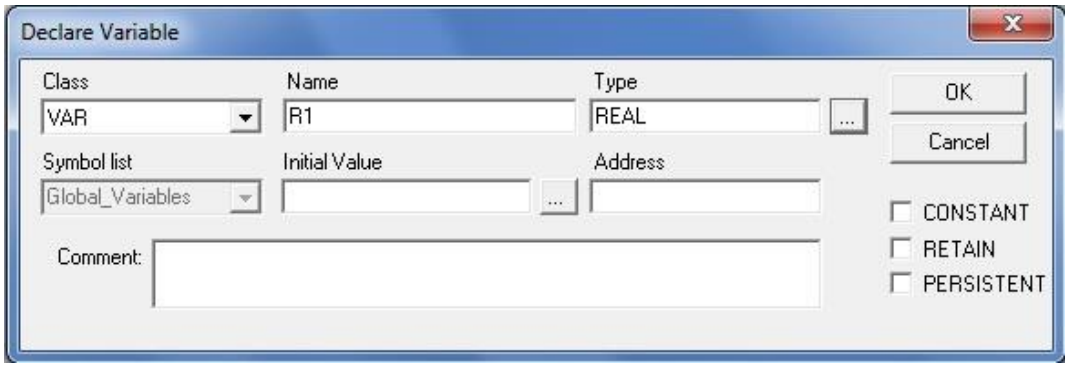

Рисунок 3.7

Аналогічно вставляємо інші арифметичні оператори та оголошуємо змінні. Нижче розташовуємо ланцюжок арифметичних операторів для обчислення другого виразу. Тепер наш програмний код має наступний вигляд (рис. 3.8):

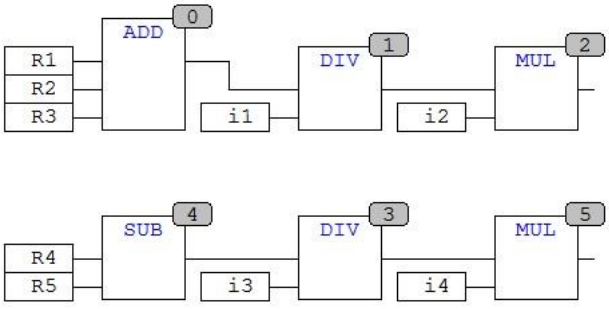

Рисунок 3.8

Тепер згідно завдання нам потрібно порівняти результат обидвох виразів, та вибрати мінімальне значення. Для цього скористаємось оператором «**MIN**». Далі це мінімальне значення треба обмежити оператором «**LIMIT**» в межах від 0 до 100. Тепер наш програмний код має наступний вигляд (рис. 3.9):

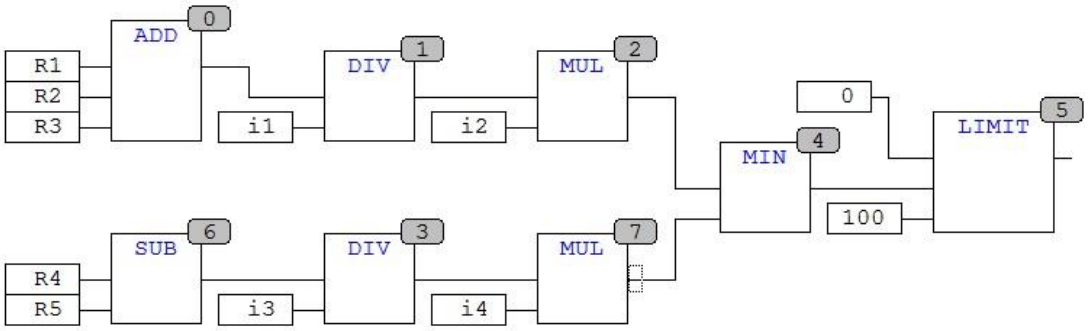

Рисунок 3.9

Далі наш результат потрібно порівняти із значенням «75». Для цього скористаємось оператором порівняння «**GT**». Якщо наш результат більший за 75 на виході оператора «**GT**» буде значення TRUE, якщо наш результат рівний або менший за 75 то на виході оператора «**GT**» буде FALSE.

Оператор «**SEL**» подає на вихід значення верхньої чи нижньої уставки в залежності від значення командного входу типу «**BOOL**».

Оператор «**MUX**» подає на вихід значення із входів в залежності від значення командного входу типу «**WORD**». Добавляємо оператор порівняння «**GT**» та оператори вибору «**SEL**» та «**MUX**». Тепер наша програма виглядає так (рис. 3.10):

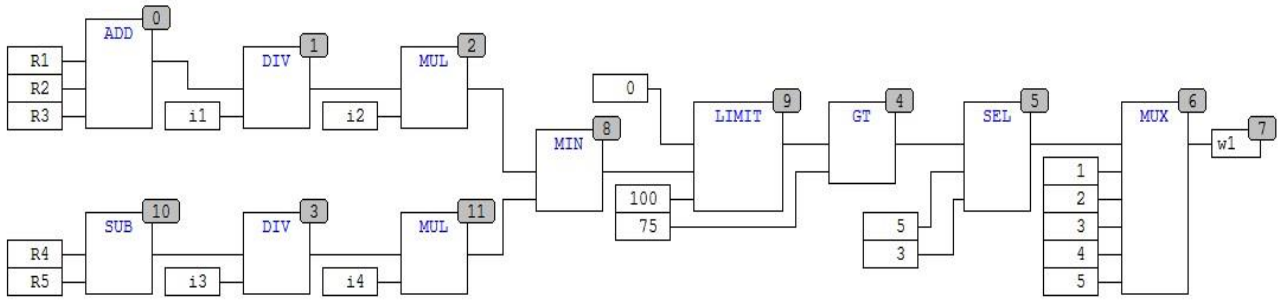

Рисунок 3.10

Як бачимо порядок виконання (цифра у верхньому кутику блоків) нашої програми невірний. Тому розставляємо порядок виконання. Клікніть ПКМ на полі програми та оберіть у контекстному меню пункт **Порядок / У відповідності із потоком данних** (**Order / Order everything according to data flow**). Тепер наша програма виглядає так (рис. 3.11):

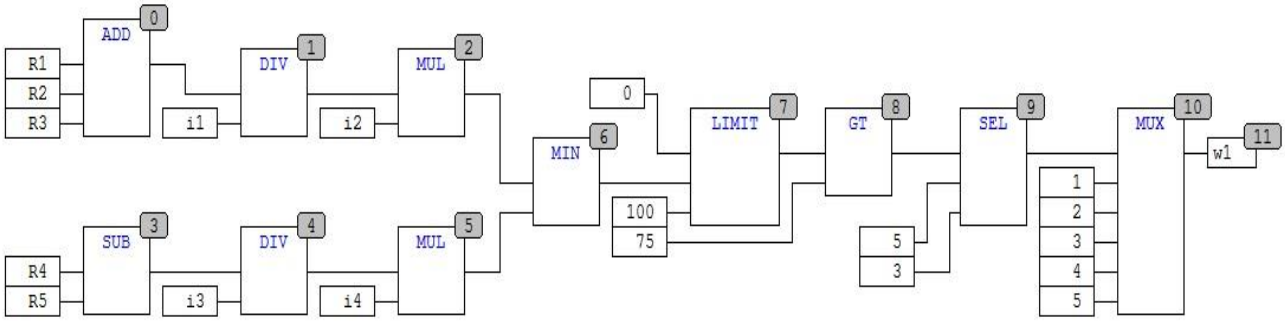

Рисунок 3.11

Перевіримо наш проект на помилки. Відкомпілюйте проект повністю командою верхнього меню «**Проект / Компилювати все**» (**Project / Rebuild**

all), або скористайтеся клавішею «**F11**». Якщо ви все виконали вірно, то в нижній частині вікна повідомлень має з'явитись надпис «0 errors» (рис. 3.12):

Рисунок 3.12

В іншому випадку необхідно виправити допущені помилки. В цьому допоможуть повідомлення про помилки.

Залишилось підставити наші значення згідно із варіантом завдання. Запускаємо проект в режимі **Емуляції (Simulation mode) - Онлайн / Підключення, Онлайн / Старт (Online / Login, Online / Run)**. Як бачимо значення усіх змінних по замовчуванні рівне нулю. При виконанні програми двічі клікніть ЛКМ на назві змінної, значення якої ви бажаєте змінити та введіть в діалоговому вікні нове значення (**New Value**) (рис. 3.13):

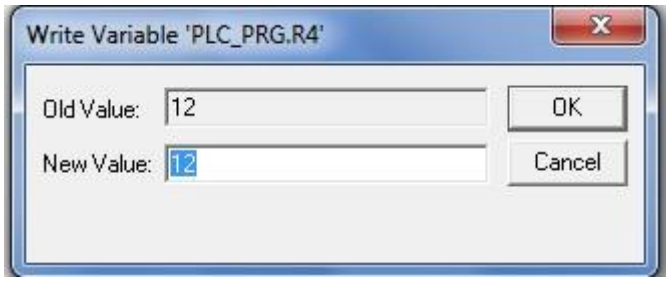

Рисунок 3.13

Як бачимо нові значення з'явились поряд із старими, проте вони не активні. Їх треба записати у програму, що вже виконується. Для цього оберіть пункт верхнього меню **Онлайн / Записати значення (Online / Write values)** або скористайтесь комбінацією гарячих клавіш **Ctrl + F7.**

Проте при відключенні від ПЛК ці значення обнуляються. Щоб значення наших змінних не вводити кожного разу, присвоїмо їм ці значення по замовчуванні. Для цього перейдіть у вікно оголошення змінних та присвойте значення змінної наступним чином:

```
PROGRAM PLC_PRG
VAR
R1: REAL:=8;
i1: INT:=9;
……..
w1: WORD;
END_VAR
```

Тепер при запуску проекту значення наших змінних відразу будуть записані у програму ПЛК.

ПРАКТИЧНА РОБОТА №4.

Тема: Вивчення середовища CodeSys для програмування ПЛК ОВЕН. Програмування на мові FBD та SFC.

Мета: Вивчити методи програмування мікроконтролерів ПЛК ОВЕН 110-60М на мовах FBD та SFC.

Ціль роботи:

1. Знайомство з основними відомостями про мову програмування FBD та SFC.
2. Знайомство із типами тригерів та таймерів.
3. Одержання навичок роботи з тригерами і таймерами.

Короткі теоретичні відомості про мову FBD

Мова FBD (Function Block Diagram, мова функціональних блоків) є графічною. Вона працює з послідовністю ланцюгів, кожен з яких містить логічний або арифметичний вираз, виклик функціонального блоку, перехід або інструкцію повернення. Найбільш важливі функції ви можете знайти в контекстному меню, яке викликається правою кнопкою миші або натисненням сполучення клавіш <Ctrl> + <F10>.

Програма на цій мові складається з функціональних блоків, з'єднаних між собою, причому блоки організовані в ланцюги, розташовані один під іншим. Спочатку виконуються зліва направо всі дії в першому зверху ланцюгові, потім – у другому і т.д. Приклад програми з одного ланцюга на мові FBD наведено на рис. 4.1. Ця програма реалізує вираз: Y=(A+B)*C

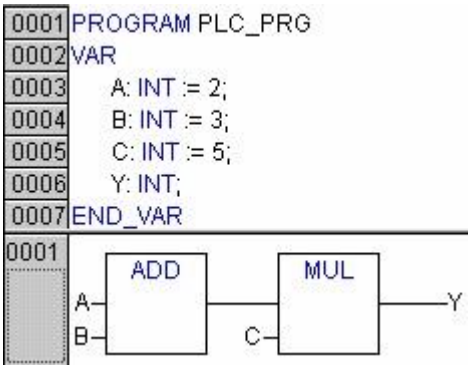

Рис. 4.1 Приклад програми на мові FBD.

Короткі теоретичні відомості про мову SFC див. у ПР №6.

Завдання для розробки.

Розробити програму (на мові FBD) для контролю оператором переміщення об’єкта. Оператор повинен періодично підтверджувати правильність переміщення об’єкта. У протилежному випадку слід видати

попередження та зупинити роботу. Об’єкт повинен здійснювати циклічний рух по периметру прямокутника (рух об’єкта описати у програмному компоненті «Machine» на мові SFC). У програмі використати тригери і таймери. У візуалізації передбачити кнопку «Пуск» для запуску програми, табло «Лічильник» для відображення кількості здійснених об’єктом циклів, кнопку підтвердження правильності переміщення об’єкта «ОК», а також сигнальні лампи: «Увага» та Стоп». Кнопка «ОК» має запускати рух об’єкта як при натисканні так і при відтисканні. Сигнальна лампа «Увага» має засвічуватися через час t_1 після натиску кнопки підтвердження «ОК». Сигнальна лампа «Стоп» має засвічуватися через час t_2 після активації лампи «Увага» та одночасно із зупинкою об’єкта (табл. 4).

Таблиця 4 – Варіанти завдань

№ п/п	Час t_1 до увімкнення лампи «Увага», с	Час t_2 до увімкнення лампи «Стоп», с	Переміщення об’єкта по горизонтальній осі до зміни напрямку	Переміщення об’єкта по вертикальній осі до зміни напрямку
1	5	5	100	50
2	6	6	80	60
3	7	7	90	70
4	8	8	110	80
5	7	9	100	60
6	6	10	120	70
7	5	11	70	80
8	4	12	100	70

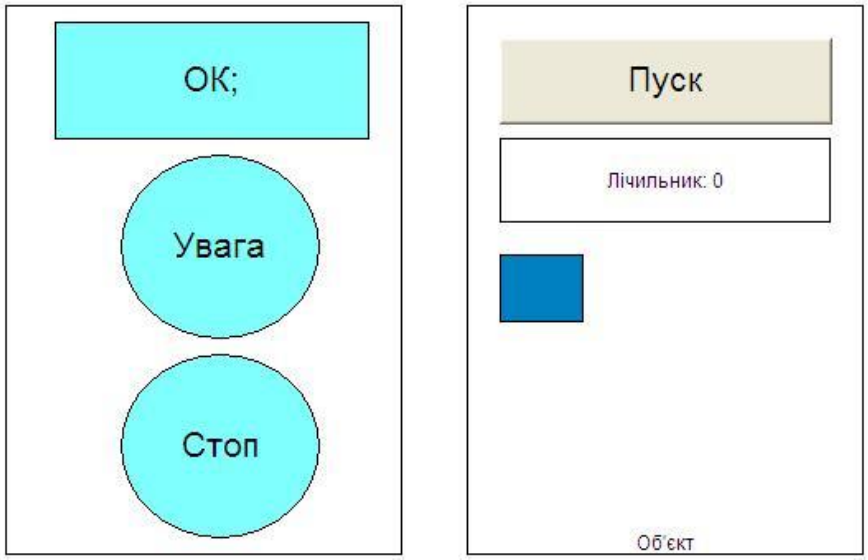

Рисунок 4.2 Загальний вигляд вікна візуалізації проекту

Порядок виконання

Створіть новий проект обравши пункт верхнього меню **File / New**. Оберіть модель мікроконтролера **ПЛК Овен 110_60М**. Натисніть **ОК**. Обираємо мову програмування для нього **FBD**, тип програмного компоненту **Program** (рис. 4.3).

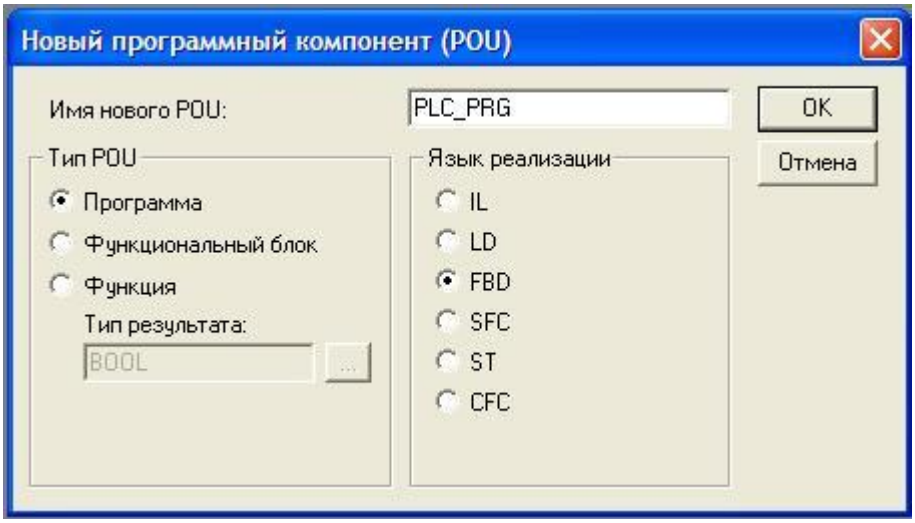

Рисунок 4.3

Оголошуємо змінну кнопки «ОК».

Змінна, що відповідає за кнопку «ОК» буде змінювати своє значення при підтвердженні коректності переміщення об'єкта оператором.

В першому ланцюжку графічного FBD редактора виділіть рядок запитань **???** та введіть назву нашої першої змінної. Нехай це буде **Observer** (наглядач). Тепер нажміть на клавіатурі стрілку вправо. З'явиться діалогове вікно оголошення змінної. В ньому вказуємо логічний тип змінної. Оскільки кнопка «ОК» може бути натиснута або відтиснута, тобто приймати значення ВКЛ/ВИКЛ нам підходить змінна типу BOOL (**Type BOOL),** яка може приймати значення TRUE або FALSE. Змініть клас змінної (**Class**) на глобальний (**VAR_GLOBAL**), оскільки дія нашої кнопки має поширюватись на обидва програмних компоненти (рис. 4.4).

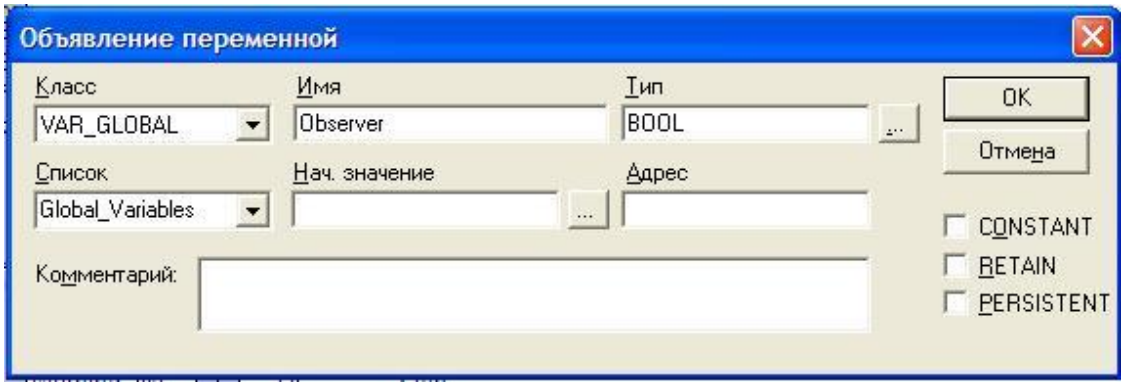

Рисунок 4.4

Натисніть **ОК** для закриття діалогового вікна. Тепер змінна **Observer** має з'явитися у вікні оголошення глобальних змінних проекту (**Global_Variables)**, що у закладці **Ресурси** (рис. 4.5).

```
VAR_GLOBAL
        Observer: BOOL;
END_VAR
```

Рисунок 4.5

Детектор переднього фронту

Оператор має підтверджувати роботу саме перемиканням кнопки «ОК», а не просто спати з постійно натиснутою кнопкою підтвердження. Щоби розділити ці ситуації треба визначити моменти натискання і відтискання, тобто переходи значень логічної змінної із нуля (FALSE) в одиницю (TRUE) і навпаки.

Повертаємось у вікно редактора програмного компоненту PLC_PRG. Виділіть курсором позицію справа від змінної **Observer.** Ви маєте побачити маленький пунктирний прямокутник. Натисніть на ньому ПКМ. В контекстному меню вводу задайте команду «**Вставити / Елемент**»(**Box**).

По замовчуванні вставляється елемент **AND**. Скористаємось асистентом вводу: натисніть клавішу **F2.** У діалоговому вікні (зліва) виберіть категорію: стандартні функціональні блоки (**Standard Function Blocks**). Із тригерів (**trigge**r) стандартної бібліотеки (**standard.lib**) виберіть **R_TRIG.** R_TRIG формує логічну одиницю по передньому фронту на вході.

Необхідно задати ім’я для нового екземпляру функціонального блоку R_TRIG. Натисніть ЛКМ над зображенням тригера та введіть ім’я **Trig1**. У діалоговому вікні оголошення змінних вкажіть клас **Class VAR** (локальні змінні), оскільки даний тригер буде діяти лише у текучому програмному компоненті. Крім того вкажіть ім’я (**Name**) **Trig1** і тип (**Type R_TRIG**) (рис. 4.6).

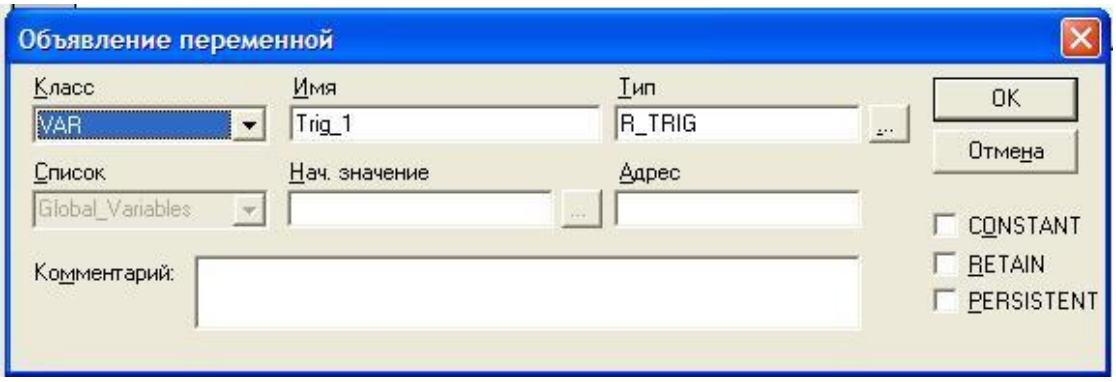

Рисунок 4.6

Натисніть **ОК** для закриття діалогового вікна. Тепер вікно нашої програми має наступний вигляд (рис. 4.7):

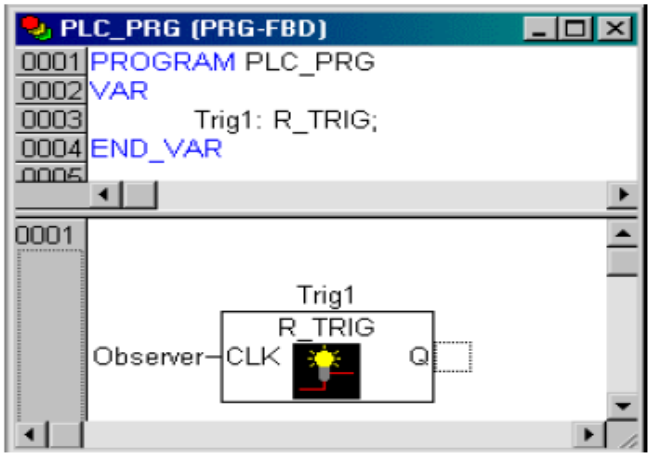

Рисунок 4.7

Детектор заднього фронту.

Виділіть вихід функціонального блоку **Trig1** і вставте як було описано вище елемент **AND**. Перейменуйте його в **OR** (логічне АБО). Виділіть вільний вхід елементу **OR** і вставте перед ним екземпляр функціонального блоку **F_TRIG** під іменем **Trig2** (рис. 4.8)**.**

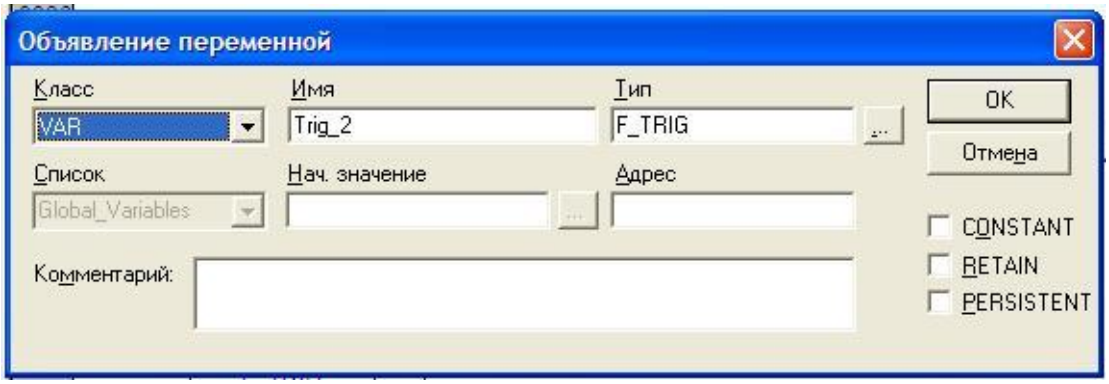

Рисунок 4.8

На вхід **F_TRIG,** з допомогою асистента вводу (F2) подайте (категорія **Global_Variables**) змінну **Observer**.

Контроль часу, перша частина.

Вставте після OR екземпляр функціонального блоку **TOF** (таймер затримки вимкнення) під іменем **Timer1.** Замініть три знаки питання на вході **PT** константою **T#10s**. Це відповідає 10 секундам. Цей час можна міняти в процесі налагодження. Ми використовуємо таймер затримки вимкнення тому, що для його роботи достатньо подати на вхід **IN** короткотривалий імпульс, котрий видає тригер (рис. 4.9).

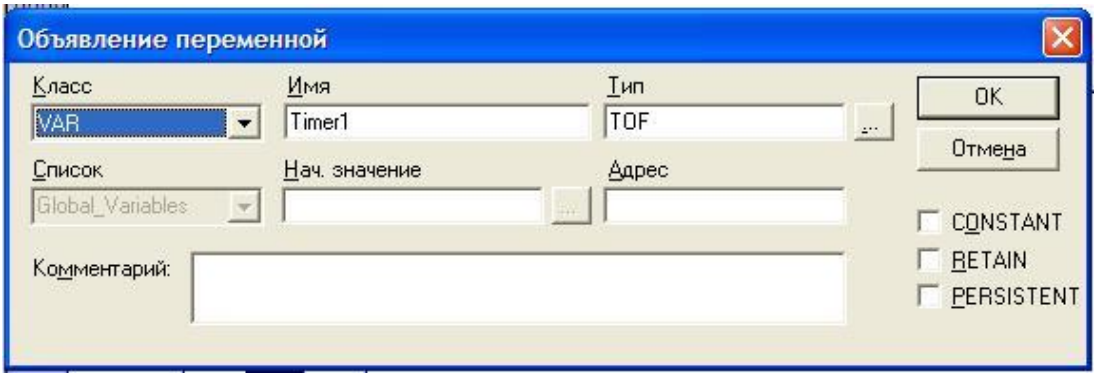

Рисунок 4.9

Вивід попередження.

Виділіть вихід Q таймера **Timer1** і в констектному меню (ПКМ) задайте команду **Assign** (присвоєння). Замініть знаки питання на ім’я змінної **Warning**. У діалоговому вікні оголошення змінних задайте клас (**Class VAR_GLOBAL)** і тип **BOOL** (рис. 4.10).

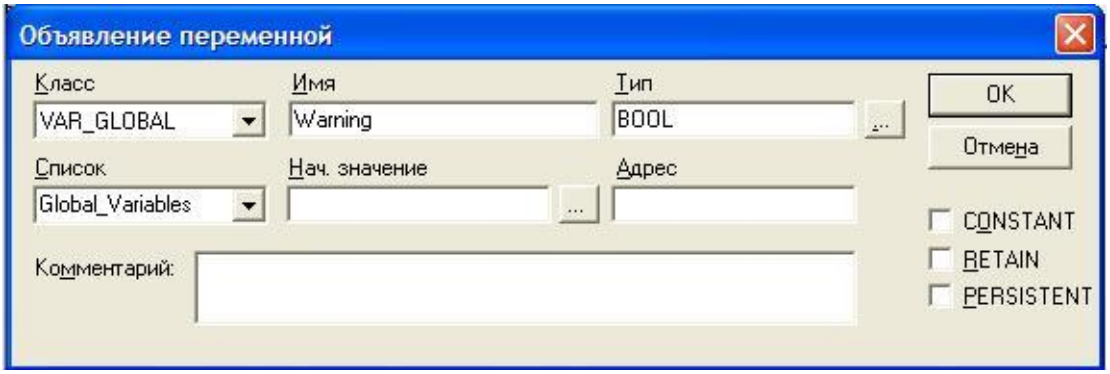

Рисунок 4.10

Тепер виділіть позицію в середині лінії, що з'єднує вихід таймера і змінну **Warning**. Задайте команду **Negate** (Інверсія) в контекстному меню. Маленький кружок означає інверсію значення логічного сигналу. Якщо все вірно, наша програма матиме такий вигляд (рис. 4.11):

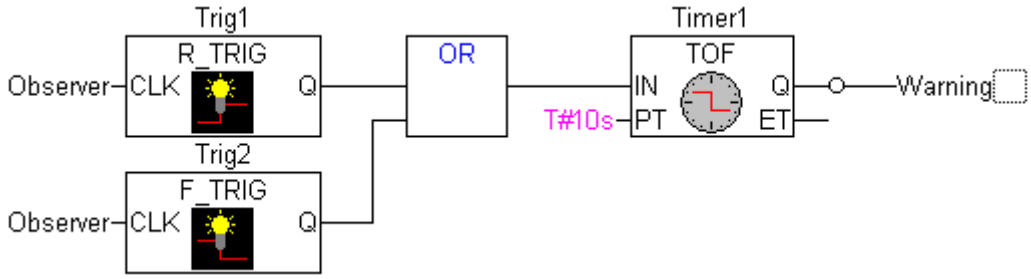

Рисунок 4.11

Формуємо стоп-сигнал по другому інтервалу часу.

Створіть новий ланцюжок FBD командою контекстного меню «**Цепь(после)**» **Insert/Network(after)**. Вставте із стандартної бібліотеки в новий ланцюжок елемент (**Box**) типу **TON** (таймер із затримкою увімкнення) під іменем **Timer2** (рис. 4.12).

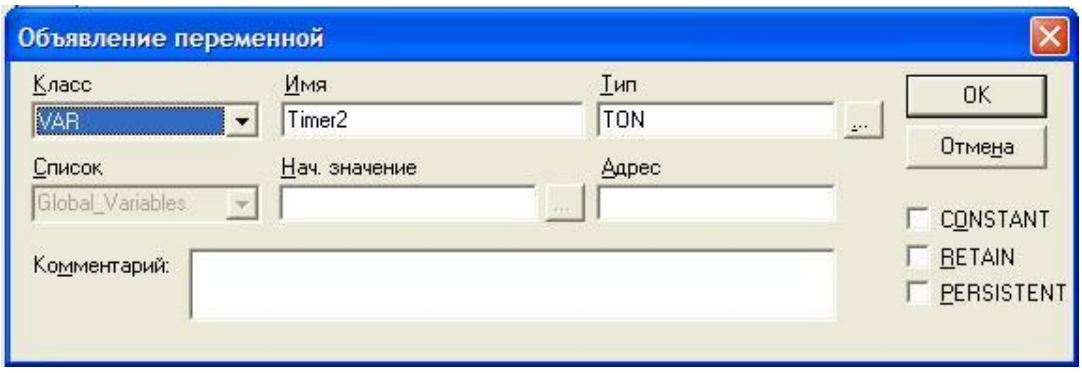

Рисунок 4.12

Подайте змінну **Warning** на вхід **IN** (використовуйте асистент вводу «F2») і константу **T#5s** на вхід **PT**. Вихід екземпляру функціонального блоку **Timer2** присвойте (**Assign**) новій глобальній логічній змінній **Stop** (рис. 4.13).

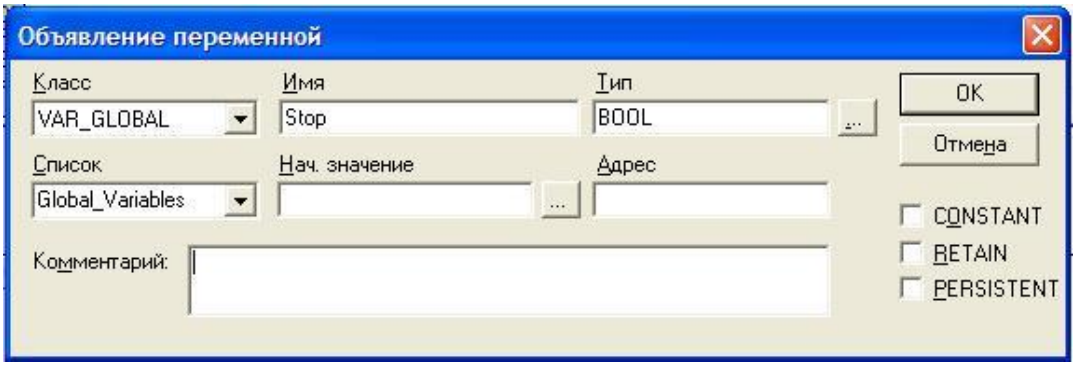

Рисунок 4.13

Програмний код у другому ланцюжку тепер матеми вигляд (рис. 4.14). Якщо все виконано вірно, вікно оголошення змінних PLC_PRG тепер матиме вигляд (рис. 4.15):

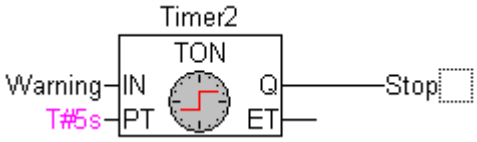

Рисунок 4.14

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     Trig_1: R_TRIG;
0004     Trig_2: F_TRIG;
0005     Timer1: TOF;
0006     Timer2: TON;
0007 END_VAR
```

Рисунок 4.15

Створюємо програмний компонент (POU) керування механізмом.

В лівій частині вікна CoDeSys розташований організатор об’єктів POUs (перша закладка, в ній присутня PLC_PRG). Вставте командою контекстного меню **Add object** (добавити об’єкт) новий програмний компонент з іменем **Machine**. Виберіть тип Програма (**Type program**) і оберіть для нього мову програмування SFC (**Language SFC**) і натисніть **ОК** (рис. 4.16).

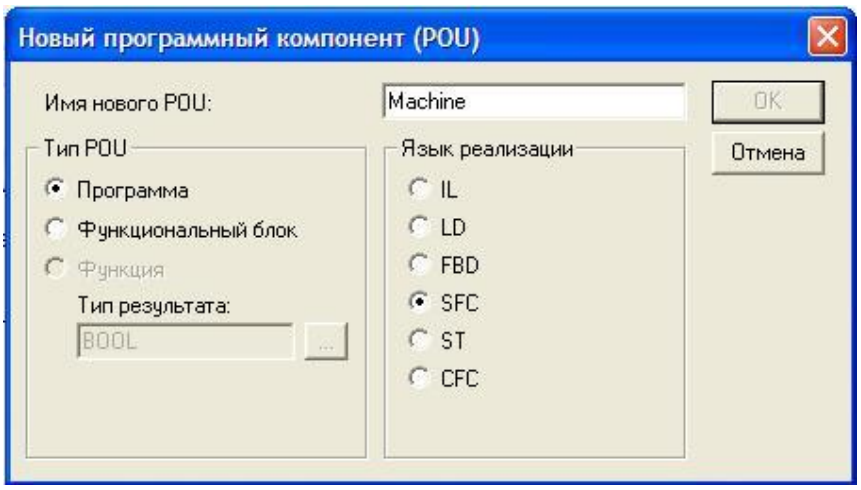

Рисунок 4.16

По замовчуванні створюється пуста діаграма, яка містить початковий крок «Init» і слідуючий перехід «Trans0», що закінчується поверненням до Init (рис. 4.17).

Задаємо послідовність роботи механізму.

Наш механізм має рухатись по замкнутій концентричній траекторії. Кожній фазі роботи має відповідати певний етап (крок). Виділіть перехід (Trans0) так, щоб він був оточений пунктирною рамкою (рис. 4.18). В контекстному меню дайте команду вставки кроку і переходу під виділенням: **Step-Transition (after)**. Аналогічно повторіть вставку кроку з переходом ще 4 рази. В нас має бути 6 кроків з переходами, включно із кроком Init (рис. 4.19).

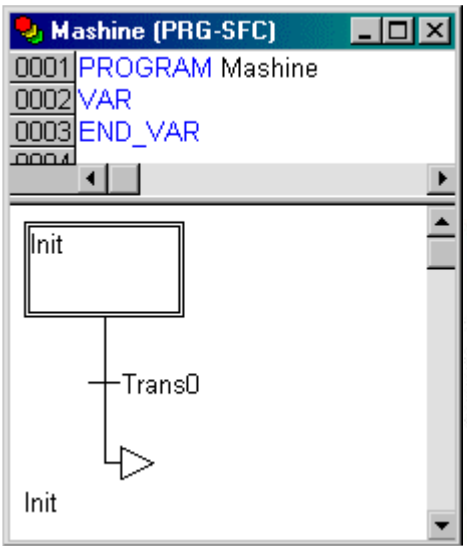

Рисунок 4.17

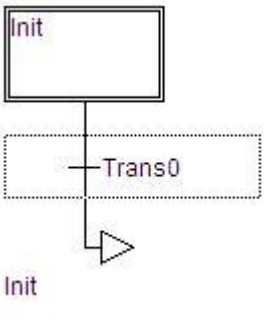

Рисунок 4.18

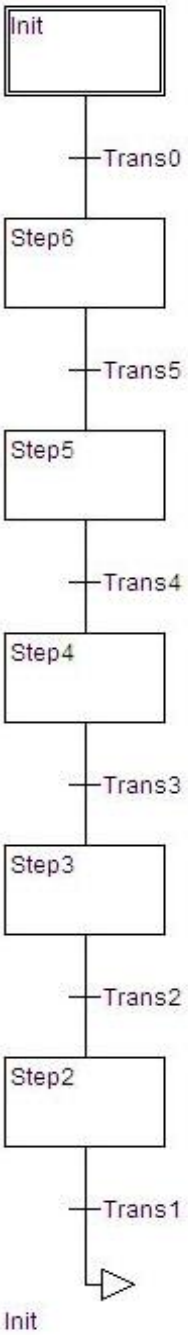

Рисунок 4.19

Клікніть ЛКМ по іменах кроків. Ви помітите, що вони виділяються кольором. Таким чином ви можете задати крокам імена.

Перший крок після Init назвіть **Go_Right**, він відповідатиме за рух механізму вправо. Наступні відповідно назвемо **Go_Down**, **Go_Left**, **Go_Up** і **Count**. Вони відповідатимуть за рух вниз, вліво та вгору відповідно. Крок **Count** відраховуватиме кількість здійснених кіл.

Програмуємо перший крок.

Клікніть двічі на кроці **Go_Right.** CoDeSys почне визначати дію кроку і попросить вибрати мову його реалізації (рис. 4.20).

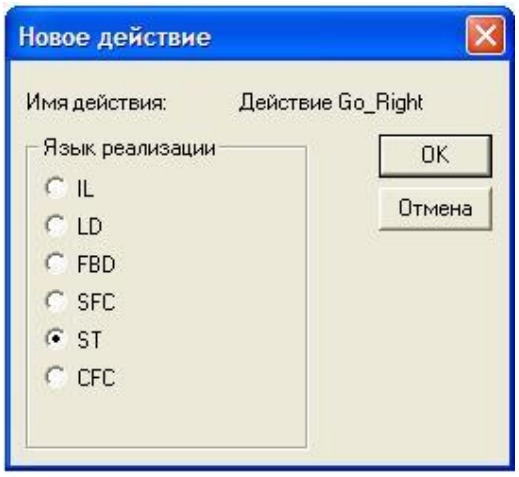

Рисунок 4.20

Виберіть мову **ST** (Structured Text) і перейдіть в автоматично відкрите вікно текстового редактора. В цьому кроці ми пишемо програмний код, що буде відповідати за рух робочого органу нашого механізму по осі X вправо. Програма має виглядати так:

X_pos:=X_pos+1;

Завершіть введення натисненням клавіші Enter. У діалоговому вікні оголошення змінних вкажіть клас **Class VAR,** та виберіть тип **INT** (цілі числа). Змінна **X_pos** відповідатиме за рух нашого механізму вправо (додатні значення) та вліво (від'ємні значення) по X.

Тепер вікно текстового редактора кроку **Go_Right** можна закрити. Тепер верхній кут кроку **Go_Right** має бути зафарбований (рис. 4.21):

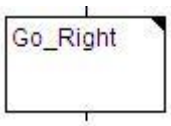

Рисунок 4.21

Це признак того, що дію цього кроку запрограмовано.

Програмуємо наступні кроки.

Повторіть описану вище послідовність для всіх кроків, що залишились. Оберіть мову **ST**. Для змінних **Y_pos** та **Counter** виберіть клас **Class VAR** і тип **INT** (рис. 4.22).

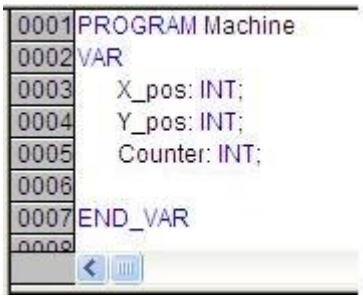

Рисунок 4.22

Програма для кроку **Go_Down** — **Y_pos:=Y_pos+1;**

Програма для кроку **Go_Left** — **X_pos:=X_pos-1;**

Програма для кроку **Go_Up** — **Y_pos:=Y_pos-1;**

Програма для кроку **Count** — **Counter:= Counter +1;**

Визначаємо умови переходів.

Перехід має містити умову, при виконанні якої програма виконує наступний крок. Перехід після кроку Init назвемо **Start**. У діалоговому вікні оголошення змінних виберіть клас **VAR_GLOBAL** і тип **BOOL** (рис. 4.23)**.**

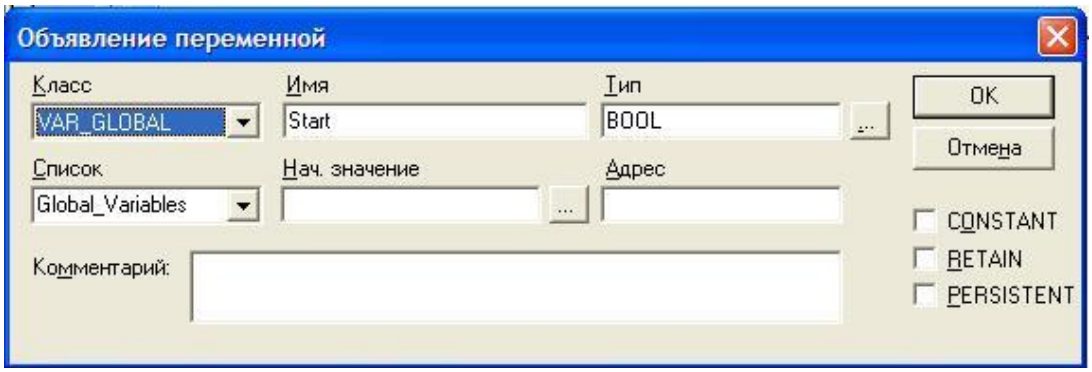

Рисунок 4.23

При одиничному значенні (TRUE) цієї змінної почнеться цикл роботи механізму.

Ми оголосили усі необхідні нам глобальні змінні. Тепер вікно глобальних змінних у закладці **Ресурси** матиме наступний вигляд (рис. 4.24):

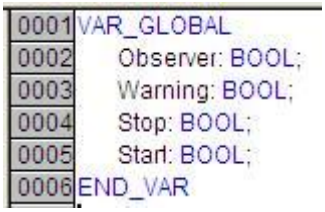

Рисунок 4.24

Наступний перехід має містити умову **X_pos = 100**. Це означає, що наш механізм буде рухатись вправо поки змінна **X_pos** не досягне значення 100, після чого дія кроку **Go_Right** припинеться, і програма перейде до виконання кроку **Go_Down.**

Умова для третього кроку **Y_pos = 50**

Умова для четвертого кроку **X_pos = 0**

Умова для п’ятого кроку **Y_pos = 0**

Умова для шостого кроку **TRUE** (тобто перехід дозволений відразу ж після однократного виконання).

Якщо все зроблено вірно наша програма **Machine** має виглядати наступним чином (рис. 4.25):

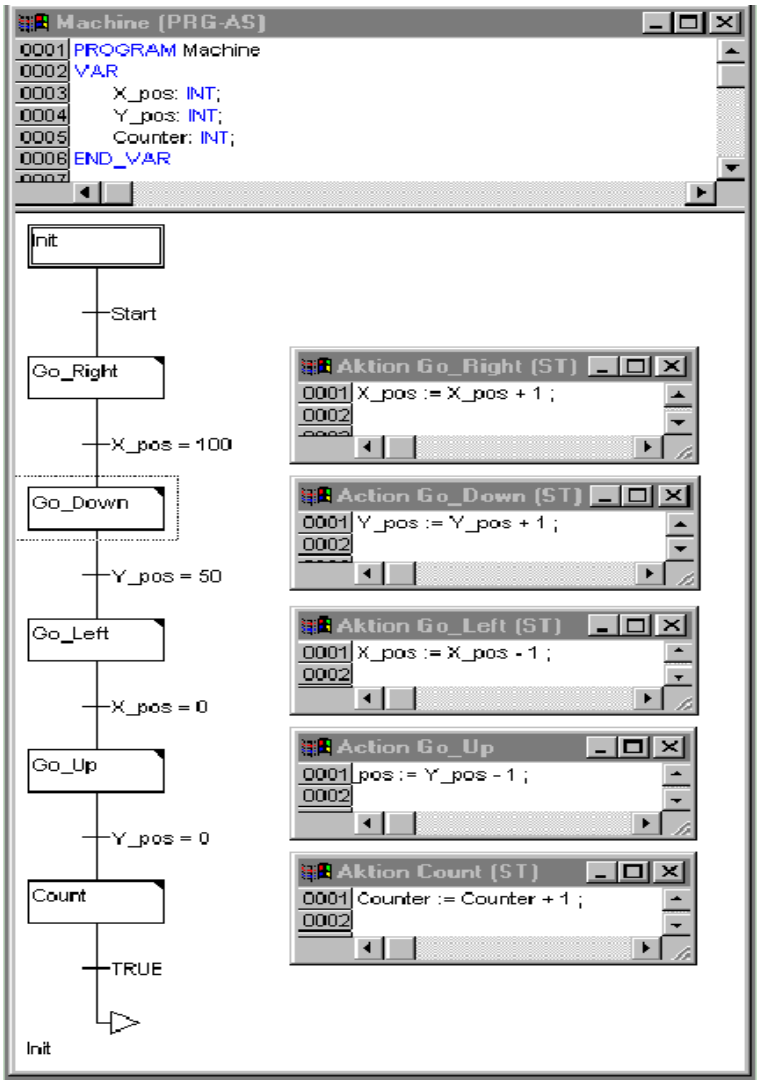

Рисунок 4.25

Зупинка механізму.

Поверніться до POU **PLC_PRG**. Добавте третій ланцюжок (Insert/Network(after). Замість знаків питання вставте змінну **Stop**. ПКМ на пунктирному прямокутнику справа від Stop і з контекстного меню вставляємо оператор **Return**. Return зупинятиме роботу програми при одиничному значенні змінної Stop (Stop = TRUE) (рис. 4.26).

Виклик POU керування механізмом.

Добавте четвертий ланцюжок (Insert/Network(after). Клікніть ПКМ на пунктирному прямокутнику і вставте елемент **Box** із контекстного меню. Як завжди, це буде **AND**. Натисніть **F2** і задайте програмний компонент керування мезанізмом (POU **Machine**) в категорії користувацьких програм (**User defined Program category**) (рис. 4.27).

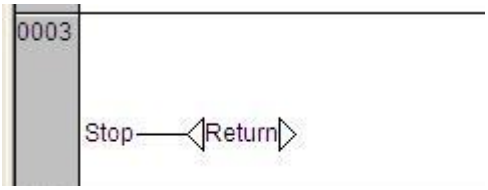

Рисунок 4.26

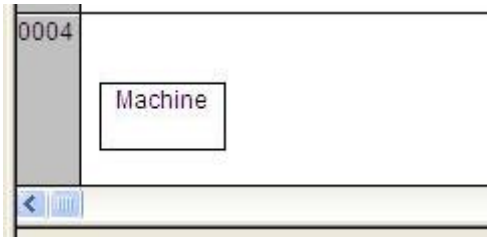

Рисунок 4.27

Компіляція проекту.

Перевіримо наш проект на помилки. Відкомпілюйте проект повністю командою верхнього меню «**Проэкт / Компилировать все**» (**Project / Rebuild all**), або скористайтеся клавішею «**F11**». Якщо ви все виконали вірно, то в нижній частині вікна повідомлень має з'явитись надпис «0 errors» (рис. 4.28)

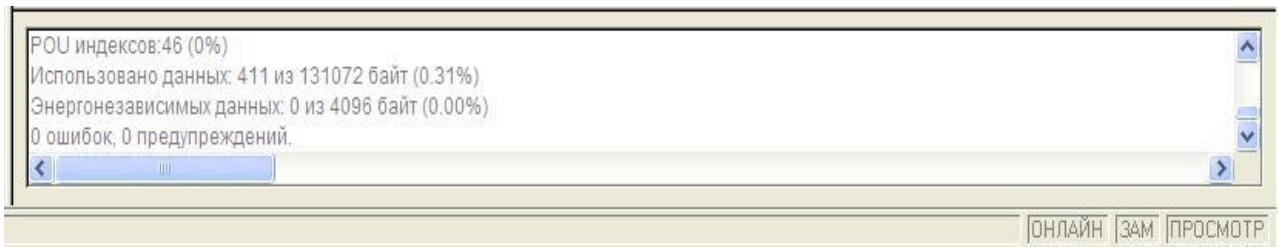

Рисунок 4.28

В іншому випадку необхідно виправити допущені помилки. В цьому допоможуть повідомлення про помилки.

Візуалізація проекту.

Добавте елемент візуалізації в третій закладці лівого вікна CoDeSys (**Vizualization / Add object**). Перейдіть на сторінку візуалізації. Присвойте новому об'єкту ім'я **Observation**. В кінці роботи вікно візуалізації має виглядати наступним чином (рис. 4.29).

На панелі інструментів виберіть команду прямокутник (**Rectangle**). У вікні редактора візуалізації натисніть ЛКМ і розтягніть прямокутник до потрібної висоти та ширини, після чого відпустіть ЛКМ.

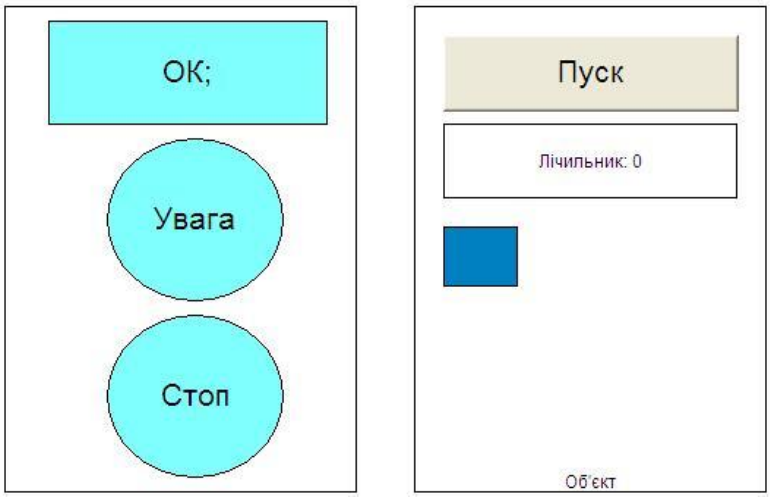

Рисунок 4.29

Налаштування першого елементу візуалізації.

Діалогове вікно налаштування елементу візуалізації викликається подвійним кліком ЛКМ на його зображенні. Задайте у вікні конфігурації елемента в категорії **Текст** слово **ОК**. Це слово відображатиметься на нашому графічному об’єкті як надпис (рис. 4.30).

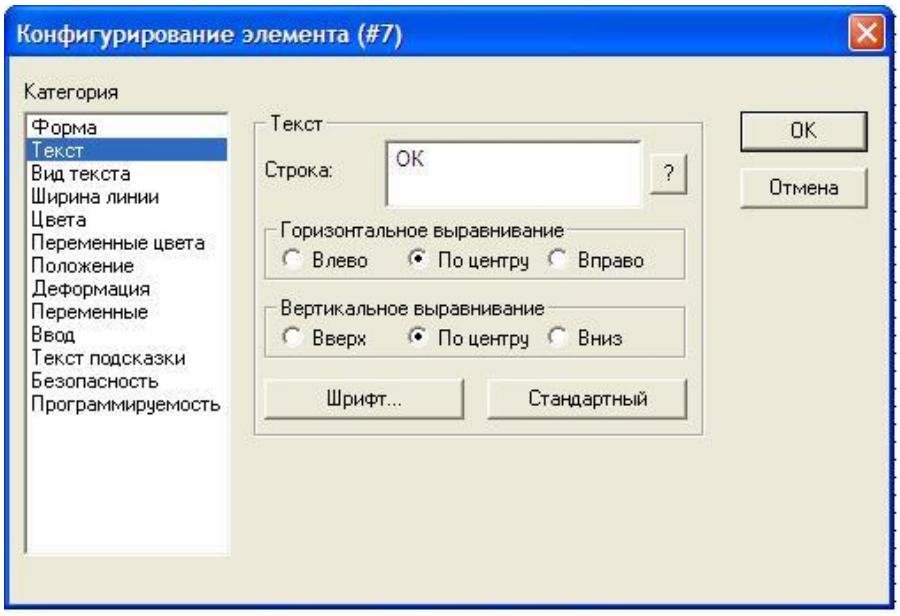

Рисунок 4.30

Тепер перейдіть в категорію змінних (**Variables**). Клікніть ЛКМ в полі зміни кольору (**Change color**) і скористуйтесь асистентом вводу «**F2**». Вставте змінну **Observer** із списку глобальних змінних. Тепер наш графічний об’єкт змінюватиме колір при зміні значення змінної **Observer** (рис. 4.31)**.**

Тепер треба задати колір який буде набувати наш прямокутник при зміні значення змінної **Observer.** Для цього перейдіть в категрію кольори **(Colors).** Задайте колір заливки елементу (**Inside**), наприклад світло-голубий. Для «збудженого» стану необхідно обрати інший колір (**Alarm color**), наприклад синій. Щоб змінна **Observer** змінювала своє значення при натисканні на наш прямокутник нам треба в категорії вводу (**Input**) ще раз ввести змінну **Observer** і поставити галочку Змінна-переключення (**Toggle variable**) (рис. 4.32).

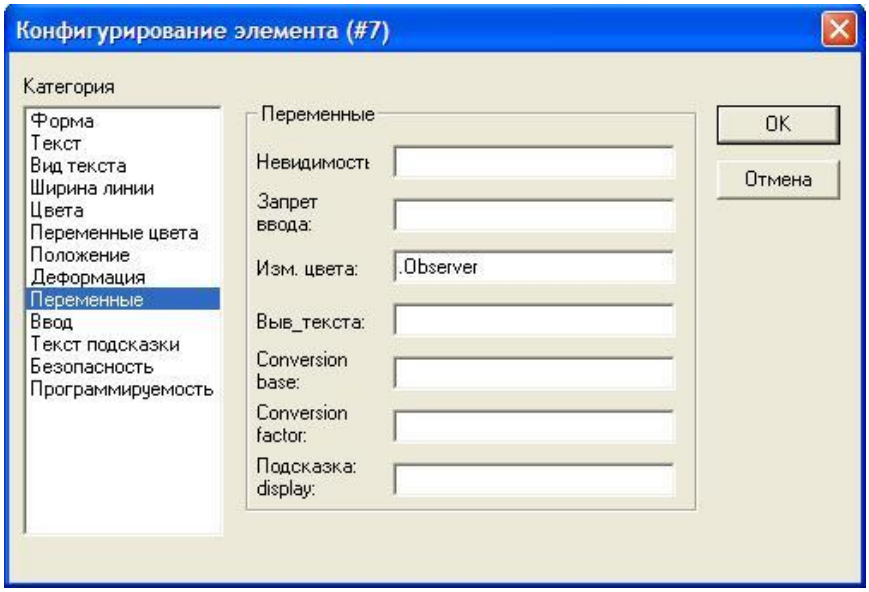

Рисунок 4.31

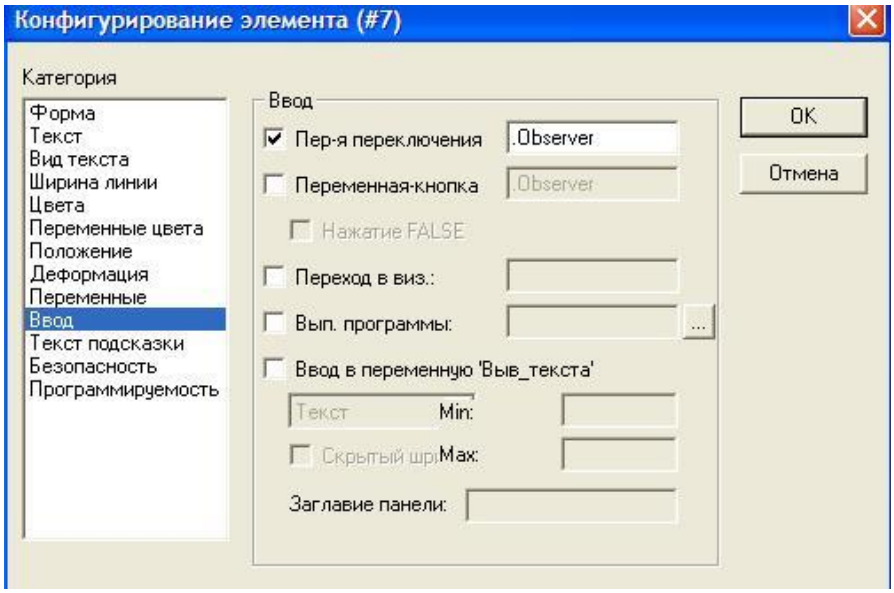

Рисунок 4.32

Тепер вікно конфігурування елементу можна закривати.

В результаті прямокутник буде відображатися світло-голубим кольором при значенні змінної **Observer** FALSE, і синім при значенні TRUE. Її значення буде змінюватися при кожному натисненні нашої клавіші **ОК**.

Налаштування решти елементів візуалізації.

Аналогічним способом побудуємо дві сигнальні лампи. Побудуйте круг (**Ellipse**). У вікні конфігурації в категорії **Текст** задайте слово «**УВАГА**». Задайте основний колір сірий, а «тривожний» - червоний. Тепер перейдіть в категорію змінних (**Variables**), і в полі зміни кольору (**Change color**) вставте змінну **Warning** із списку глобальних змінних. Закрийте вікно конфігурування елемента. Скопіюйте створений круг командою **Edit / Copy** або з використанням комбінації гарячих клавіш **Ctrl + C**. Вставте скопійований круг один раз командою **Edit / Paste** (або **Ctrl + V**). Відредагуйте конфігурацію нового круга: змініть текст «УВАГА» на слово «СТОП», а в категорії змінних в полі зміни кольору змініть змінну **Warning** на глобальну змінну **Stop.**

Побудуйте кнопку «**Пуск**» за допомогою команди **Кнопка (Button)** на панелі інструментів. Задайте наступні налаштування:

- Категорія Текст — **Пуск**;
- Категорія Ввід — галочку на змінна переключення (**Toggle variable**), в полі змінна **.Start**;

Побудуйте прямокутник для лічильника із наступними налаштуваннями:

- Категорія Текст — **Лічильник: %s** (%s означає, що значення змінної **Counter** буде відображатися у вигляді рядка символів) (рис. 4.33);
-

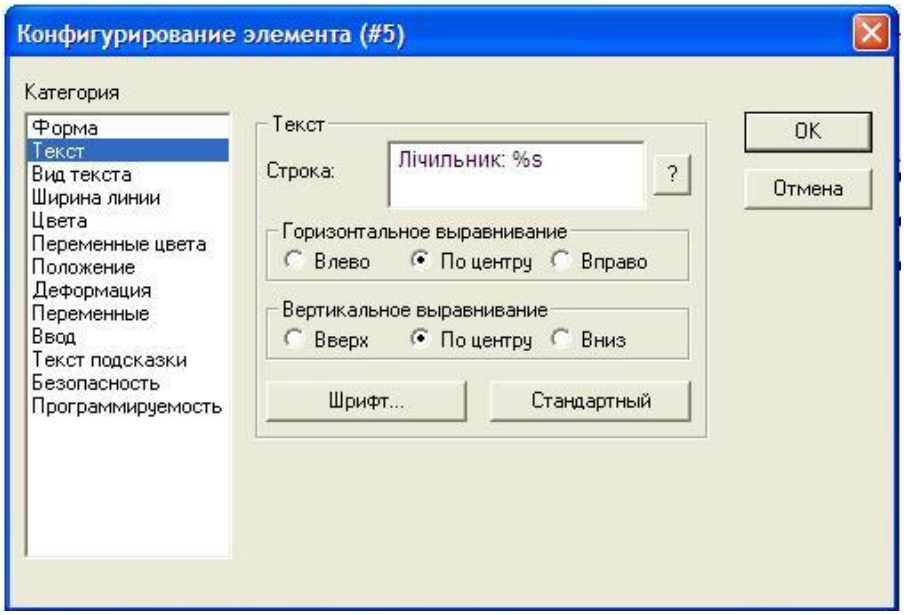

Рисунок 4.33

- Категорія Змінні — в полі **Виведення тексту** (**Text display**) змінна **Machine.Counter** (рис. 4.34)

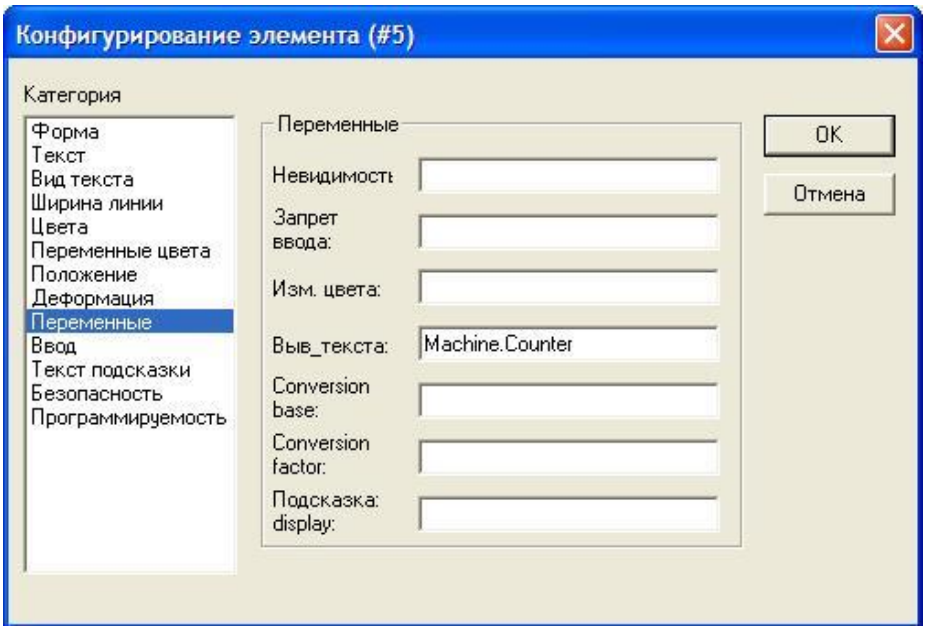

Рисунок 4.34

Побудуйте невеличкий прямокутник, що відображатиме робочий інструмент механізму, із наступними налаштуваннями:

- Категорія **Положення (Absolute movement Category) — Зсув по X (X-Offset)** змінна **Machine.X_pos**
- Категорія **Положення (Absolute movement Category) — Зсув по Y (Y-Offset)** змінна **Machine.Y_pos** (рис. 4.35)
- Категорія **Кольора (Colors)** — залийте прямокутник синім кольором.

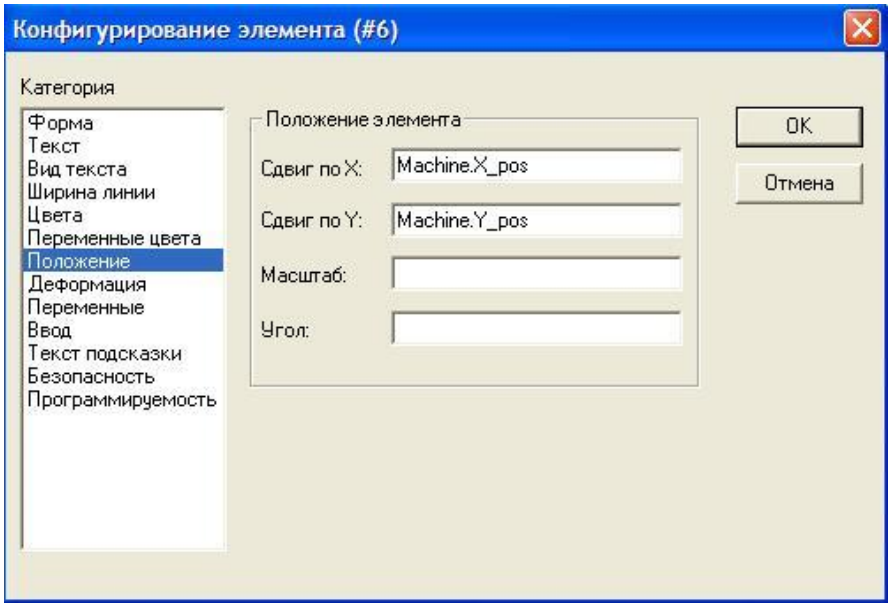

Рисунок 4.35

Нарисуйте дві декоративних рамки для розділу областей контролю і механізму. Задайте в них відповідні надписи, як вказано в завданні до лабораторної роботи, з вирівнюванням по низу (**Vertical alignment bottom**). Використовуючи контекстне меню, помістіть декоративні прямокутники на задній план (**Send to back**).

Програма має працювати наступним чином: при запуску в режимі емуляції **Онлайн / Підключення, Онлайн / Старт (Online / Login, Online / Run)** має горіти лампа «УВАГА». При натисненні на кнопку «Пуск» наш прямокутник, що відображає робочий інструмент механізму, починає рухатись по замкнутій траєкторії. Через певний час загорається лампа «СТОП» і механізм зупиняється. Оператор підтверджує правильність руху механізму натискаючи клавішу «ОК», лампа «УВАГА» і «СТОП» гаснуть, механізм продовжує рухатись. Через деякий заданий час загорається лампа попередження «УВАГА», а потім загорається лампа «СТОП» і механізм знову зупиняється. Оператор знову тисне «ОК» і все повторюється. Лічильник відраховує кількість здійснених циклів руху механізму і виводить значення на екран.

ПРАКТИЧНА РОБОТА №5

***Тема*:** Вивчення середовища CodeSys для програмування ПЛК ОВЕН.

Програмування на мовах FBD, SFC, CFC та IL.

Мета: Вивчити методи програмування мікроконтролерів ПЛК ОВЕН 110-60М на мовах FBD, SFC, CFC та IL.

Ціль роботи:

1. Знайомство з основними відомостями про мови програмування SFC та IL, CFC та FBD.
2. Одержання навичок роботи з програмними компонентами.

Короткі теоретичні відомості про мову IL

Мова IL (Instruction List, список інструкцій) – текстова мова програмування, асемблер низького рівня. Програма на цій мові являє собою список послідовних інструкцій, кожна з яких записується в окремому рядку. Приклад програми на мові IL наведено на рис. 5.1, дана програма може додавати два цілих числа. Ця мова досить популярна, але призначена в першу чергу для професійних програмістів і розробників.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     A: INT := 0;
0004 END_VAR
0001 LABEL1:
0002     LD    2 (*A=2+3*)
0003     ADD   3
0004     ST    A
```

Рисунок 5.1 Приклад програми на мові IL

В IL (Instruction list) кожна інструкція починається з нового рядка і містить оператор і, в залежності від типу операції, один і більше операндів, розділених комами. Перед операндом може перебувати мітка, що закінчується двокрапкою (:). Коментар повинен бути останнім елементом в рядку. Між інструкціями можуть перебувати порожні рядки.

В IL можна використовувати наступні оператори і модифікатори:

Модифікатори:

C з JMP, CAL, RET: інструкція виконується тільки тоді, коли результат акумулятора ІСТИНА.

N з JMPC,CALC, RETC: інструкція виконується тоді, коли результат акумулятора БРЕХНЯ.

N в інших випадках: заперечення операнда.

Нижче наведена таблиця всіх операторів IL з поясненнями і допустимими модифікаторами:

Оператор	Модифікатор	Значення
LD	N	Присвоєння акумулятора значення оператора
ST	N	Присвоєння значення акумулятора операнду

S		Присвоїти логічного операнду значення ІСТИНА, якщо значення акумулятора ІСТИНА
R		Присвоїти логічного операнду значення БРЕХНЯ
AND	N, (	Побітне І
OR	N, (	Побітне АБО
XOR	N, (	Побітне виключаюче АБО
ADD	(	Додавання
SUB	(	Віднімання
MUL	(	Множення
DTV	(	Ділення
GT	(	>
GE	(	> =
QE	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	Перехід до мітки
CAL	CN	Виклик функціонального блоку
RET	CN	Вихід з POU і повернення в зухвалу програму.
)		Обчислення затриманої операції

Після оператора можна поставити дужки, тоді значення виразу всередині дужок розглядається як операнд.

Наприклад:

```
LD      2
MUL     2
ADD     3
ST      ERG
```

Тут значення ERG дорівнює 7. Якщо поставити дужки, то порядок обчислень зміниться:

```
LD      2
MUL     (2
ADD     3
)
ST      ERG
```

Тепер значення змінної ERG дорівнює 10.

Операція MUL виконується тільки тоді, коли програма доходить до ")". В якості операнда MUL використовує значення 5.

Завдання для розробки.

Розробити програму емуляції роботи двох світлофорів. Програмний компонент увімкнення ламп світлофорів написати на мові FBD. Програмний компонент таймера роботи ламп написати на мові IL. Програмний компонент переключення режимів роботи світлофорів написати на мові SFC, кроки цього програмного компонента запрограмувати на мові IL. Запуск виконання програми описати на мові CFC (табл. 5).

Таблиця 5 Варіанти завдань

№ п/п	Кількість циклів до вимкнення світлофорів
1	5
2	6
3	7
4	8
5	9
6	10
7	11
8	12

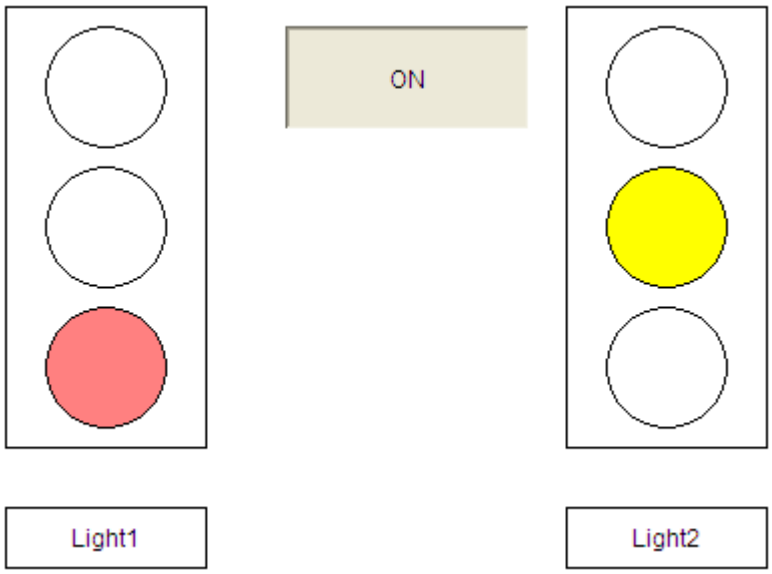

Рисунок 5.2 Загальний вигляд вікна візуалізації проекту

Порядок виконання.

Створіть новий проект обравши пункт верхнього меню **File / New.** Оберіть модель мікроконтролера **ПЛК Овен 110_60М**. Натисніть **ОК**. Не змінюйте назву нового проекту **PLC_PRG**. Обираємо мову програмування для нього **CFC**, тип програмного компоненту **Program** (рис. 5.3).

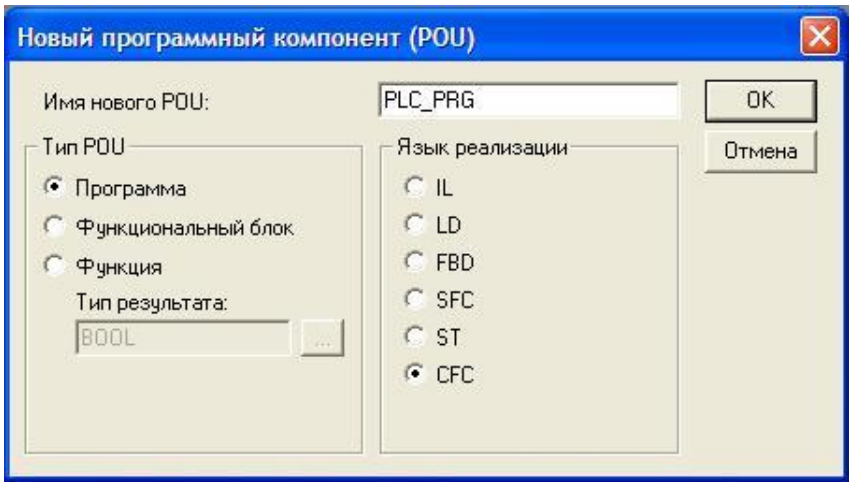

Рисунок 5.3

Тепер створимо ще три програмних компоненти. Скористайтесь командою контексного меню в організаторі об’єктів, або пунктом верхнього меню **Проект / Об’єкт / Добавити** (**Project / Object / Add**). Створіть: програму на мові SFC з ім’ям **SEQUENCE** (рис. 5.4).

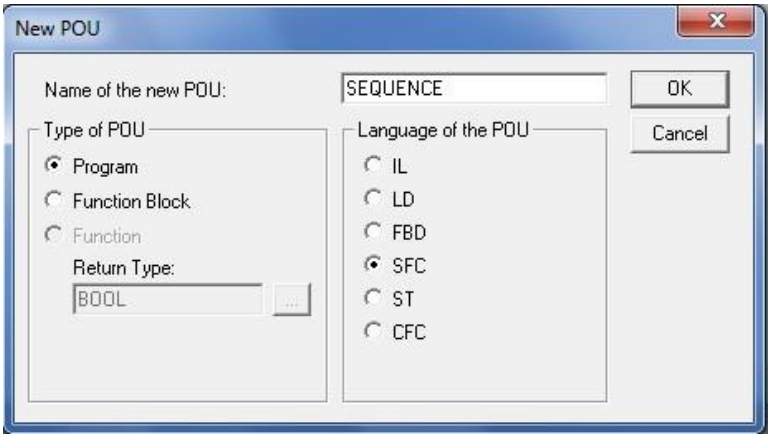

Рисунок 5.4

функціональний блок на мові FBD з ім’ям **TRAFFICSIGNAL** (рис. 5.5):

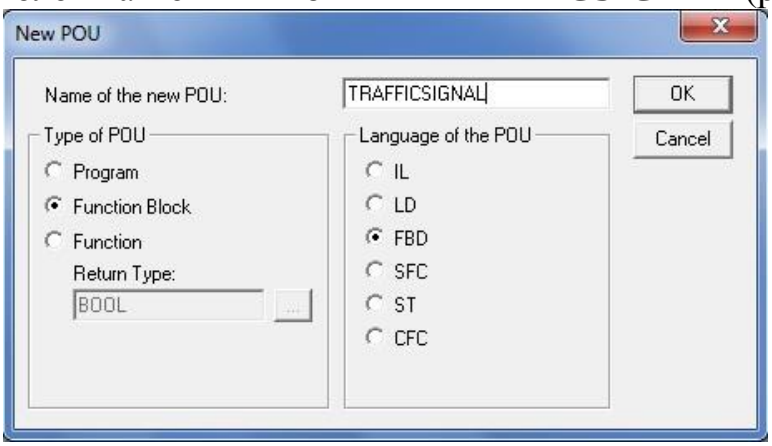

Рисунок 5.5

і ще один аналогічний блок – **WAIT** який ми будемо описувати на мові IL (рис. 5.6):

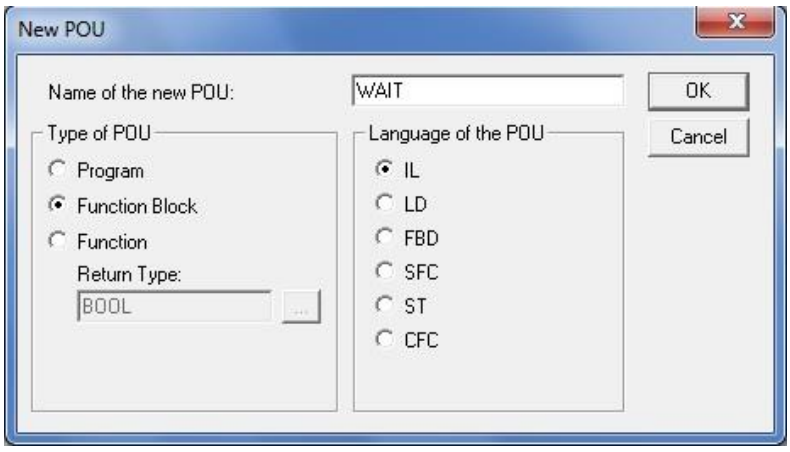

Рисунок 5.6

Тепер наш організатор об’єктів має наступний вигляд (рис. 5.7):

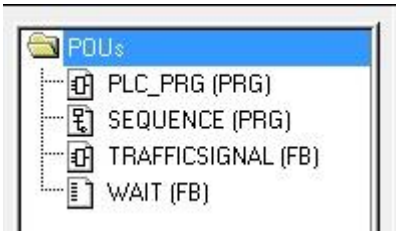

Рисунок 5.7

Що робить **TRAFFICSIGNAL**?

У POU TRAFFICSIGNAL ми зіставимо певні стадії процесу відповідним кольорам. Тобто ми переконаємось, що червоне світло запалено в червоній стадії і в жовто-червоній стадії, жовте світло в жовтій і жовто-червоній стадії і т.д.

Що робить **WAIT**?

У WAIT ми створимо простий таймер, який на вхід отримує тривалість стадії в мілісекундах і на виході видає стан ІСТИНА після закінчення заданого періоду часу.

Що робить **SEQUENCE**?

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

Що робить **PLC_PRG**?

У PLC_PRG вводиться вхідний сигнал включення, що дозволяє почати роботу світлофора і 'Команди кольорів' кожної лампи пов'язані з відповідними виходами апаратури.

Повернемося тепер до POU **TRAFFICSIGNAL**. У редакторі оголошень визначте вхідну змінну (між ключовими словами **VAR_INPUT** і **END_VAR**) на ім'я **STATUS** типу **INT**. **STATUS** матиме чотири можливих стани, що визначають відповідні стадії - зелена, жовта, жовто-червона і червона.

Оскільки наш блок **TRAFFICSIGNAL** має три виходи, потрібно визначити ще три змінних типу BOOL: **RED**, **YELLOW** і **GREEN**. Тепер розділ оголошень блоку **TRAFFICSIGNAL** повинен виглядати так (рис. 5.8):

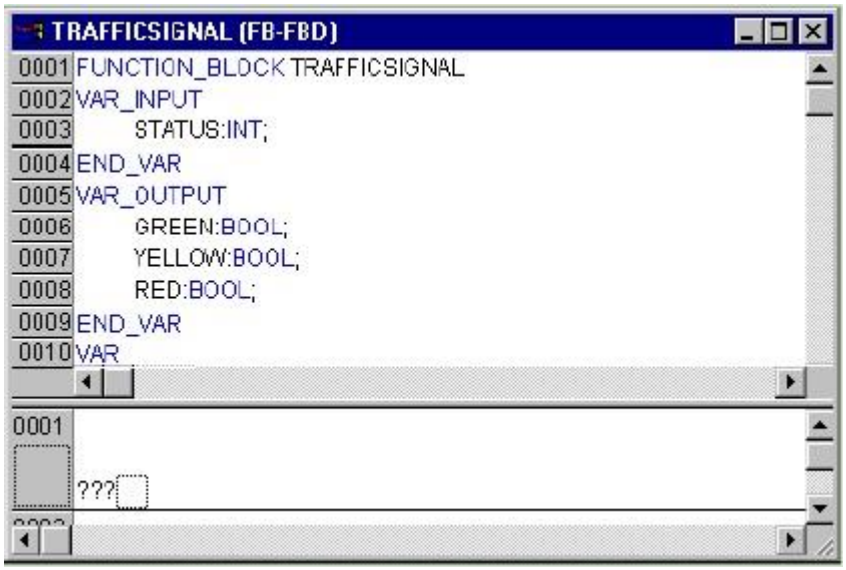

Рисунок 5.8

Програмуємо "TRAFFICSIGNAL"

Тепер пора описати, що пов'язує вхід STATUS з вихідними змінними. Для цього перейдіть в розділ коду POU (body). Клацніть на полі зліва від першого ланцюжка (сіра область з номером 1). Ви тепер обрали перший ланцюжок. Тепер дайте команду меню **Вставка / Елемент (Insert / Box)**.

У першому ланцюжку буде вставлений прямокутник з оператором AND і двома входами (рис. 5.9). Клікніть ЛКМ на тексті AND і замініть його на **EQ** (оператор порівняння). Три знаки запитання біля верхнього з двох входів замініть на ім'я змінної **STATUS**. Для нижнього входу замість трьох знаків питання потрібно поставити **1.** В результаті Ви повинні отримати наступний елемент (рис. 5.10). Клацніть тепер на місці позаду прямокутника EQ. Тепер вибрано вихід EQ. Виконайте команду меню **Вставка / Присвоєння (Insert / Assign)**. Змініть три питання **???** на **GREEN**. Тепер наш ланцюжок має наступний вигляд (рис. 5.11):

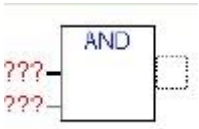

Рисунок 5.9

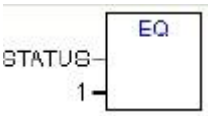

Рисунок 5.10

Рисунок 5.11

STATUS порівнюється з **1**, результат присвоюється **GREEN**. Таким чином, **GREEN** (зелене світло світлофору) буде увімкнено, коли **STATUS** дорівнює **1.**

Для інших кольорів (режимів роботи) TRAFFICSIGNAL нам знадобляться ще три ланцюжки. Створіть їх командою **Вставка / Ланцюжок (після) (Insert / Network (after))**. Закінчений POU повинен виглядати таким чином (рис. 5.12).

Щоб вставити оператор перед входом іншого оператора, Ви повинні виділити сам вхід, а не текст (виділяється пунктирним прямокутником). Далі використовуйте команду **Вставка / Елемент (Insert / Box)**.

Тепер наш перший POU закінчений. Як і планувалося, **TRAFFICSIGNAL** буде керувати активацією виходів, керуючись значенням змінної **STATUS**.

Оголошення "WAIT".

Тепер давайте повернемося до POU **WAIT**. Як передбачалося, цей POU буде працювати таймером, що буде задавати тривалість стадій **TRAFFICSIGNAL** (тобто скільки часу буде увімкнено зелене світло, скільки жовте, і т.д.). Наш POU повинен мати вхідну змінну з ім’ям **TIME** типу TIME і генерувати на виході двійкову (типу BOOL) змінну, яку ми назвемо **OK**. Дана змінна повинна приймати значення TRUE, коли бажаний період часу закінчений. Попередньо ми встановлюємо цю змінну в FALSE в кінці рядка оголошення (але до крапки з комою) ": = FALSE".

Тепер нам потрібен генератор часу (таймер типу TP). Він має два входи (IN, PT) і два виходи (Q, ET). TP робить наступне: поки IN встановлений в FALSE, ET буде рівне 0 і Q буде FALSE. Як тільки IN переключиться в TRUE, вихід ET почне відраховувати час в мілісекундах. Коли ET досягне значення заданого PT, рахунок буде зупинений. Тим часом вихід Q дорівнює TRUE, поки

ET менше PT. Як тільки ET досягне значення PT, вихід Q знову переключиться в FALSE.

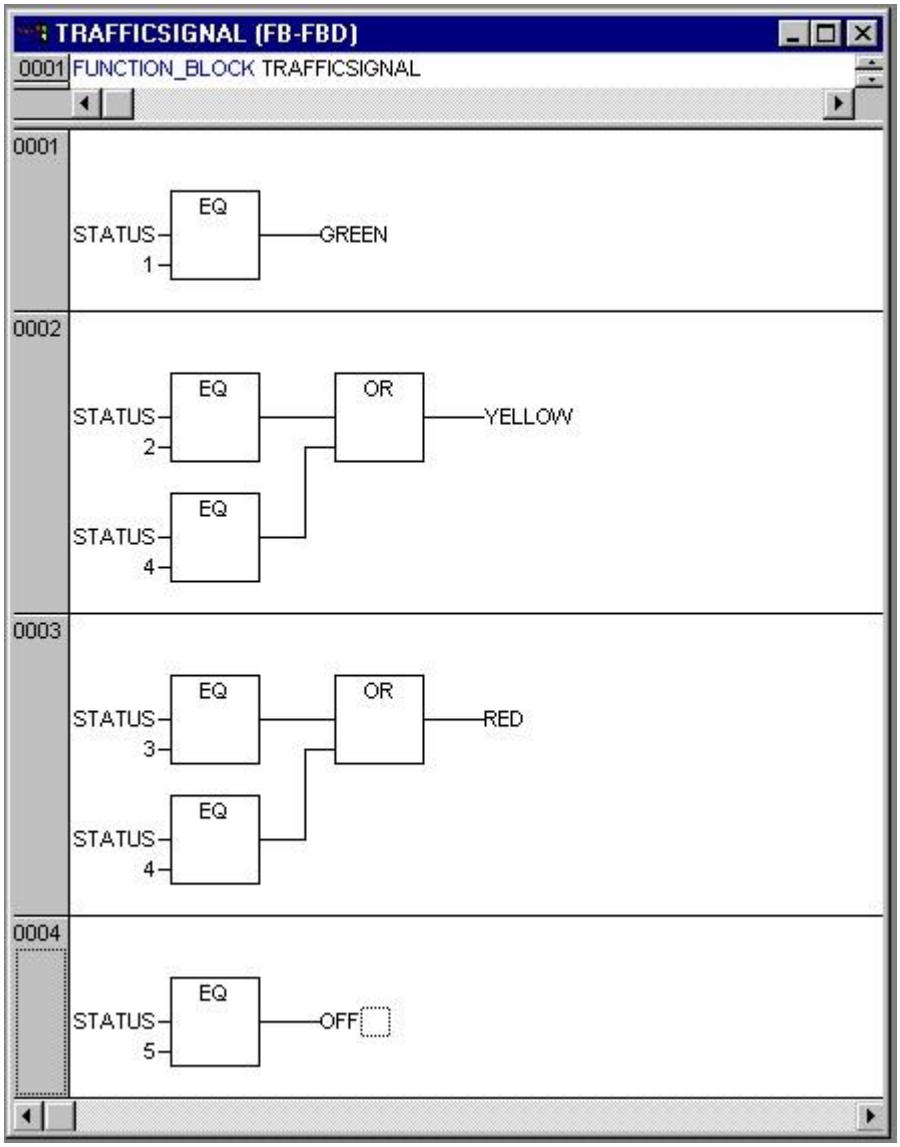

Рисунок 5.12

Щоб використовувати TP в POU **WAIT**, ми повинні створити його локальний екземпляр. Для цього ми оголошуємо локальну змінну з ім'ям **ZAB** (відрахований час) типу TP (між ключовими словами **VAR** та **END_VAR**).

Розділ оголошень **WAIT** тепер повинен виглядати так (рис. 5.13):

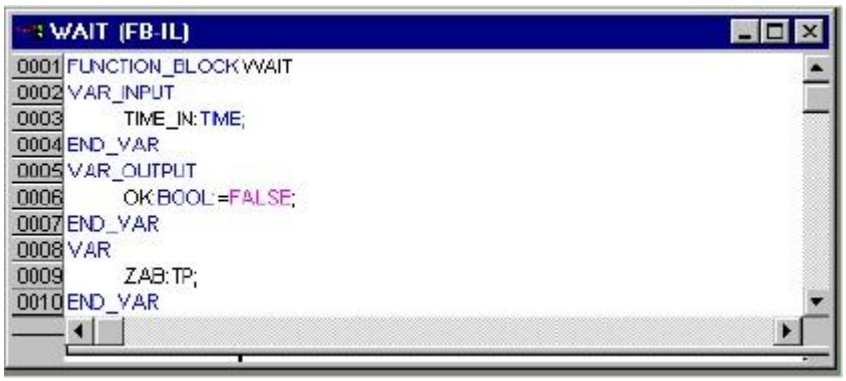

Рисунок 5.13

Програмуємо "WAIT".

Для отримання бажаного алгоритму роботи таймера текст програми на мові IL має виглядати наступним чином (рис. 5.14):

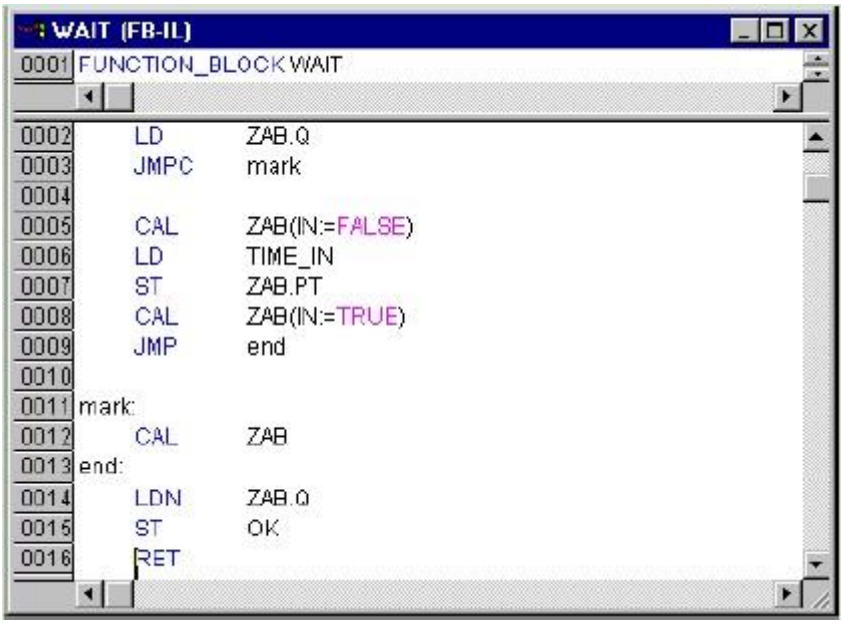

Рисунок 5.14

Працює цей програмний код наступним чином:

Спочатку перевіряється, чи встановлений Q в TRUE (можливо, відлік вже запущений), в цьому випадку ми не чіпаємо установки таймера ZAB, а викликаємо функціональний блок ZAB без вхідних змінних - щоб перевірити, чи закінчений період часу. В іншому випадку ми встановлюємо змінну IN таймера ZAB в FALSE і одночасно ET в 0 і Q в FALSE. Таким чином, всі змінні встановлені в початковий стан. Тепер ми встановлюємо необхідний час TIME змінної PT і викликаємо ZAB з IN: = TRUE. Функціональний блок ZAB тепер буде працювати, поки не досягне значення TIME і не встановить Q в FALSE. Інвертоване значення Q буде зберігатися у змінній OK після кожного виконання WAIT. Як тільки Q стане FALSE, OK прийме значення TRUE.

З таймером ми розібралися. Тепер можна об'єднати наші два блоки **WAIT** і **TRAFFICSIGNAL** в головній програмі **PLC_PRG**. Але спочатку запрограмуємо **SEQUENCE.**

Програмуємо SEQUENCE.

Спочатку оголосимо необхідні змінні. Це вхідна змінна START типу BOOL, дві вихідних змінні **TRAFFICSIGNAL1** і **TRAFFICSIGNAL2** типу INT і одна змінна типу WAIT (**DELAY**). Вікно оголошення змінних SEQUENCE тепер виглядає так (рис. 5.15):

Створюємо SFC діаграму.

Як ми пригадуємо із ПР№4 спочатку SFC діаграма завжди складається з етапу "Init" переходу "Trans0" і повернення назад до Init. Звичайно ж, нам доведеться дещо доповнити її. Перш ніж програмувати конкретні етапи і переходи, вибудуємо структуру діаграми. Спочатку нам знадобляться етапи для кожної стадії **TRAFFICSIGNAL**. Вставте їх, виділяючи Trans0 і вибираючи команди **Вставка / Крок-перехід (знизу) (Insert / Step transition (after))**. Повторіть цю процедуру ще три рази.

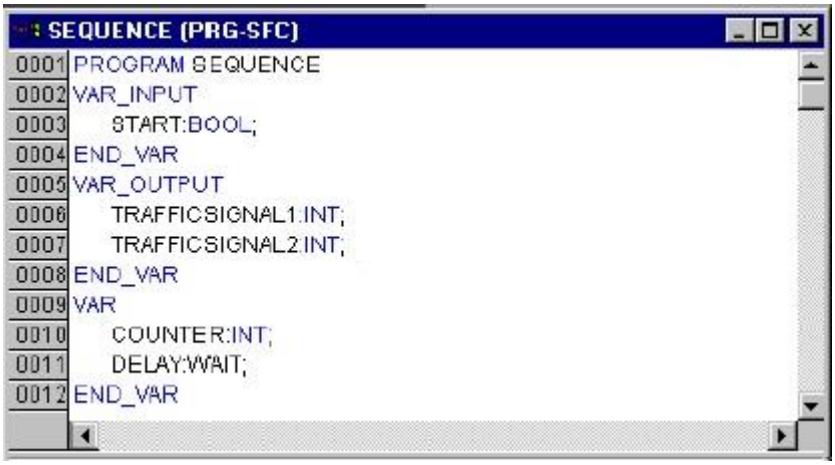

Рисунок 5.15

Для редагування назви переходу або етапу потрібно просто клікнути ЛКМ на потрібному тексті. Назвіть перший перехід після Init **START**, а всі інші переходи **DELAY.OK**.

Перший перехід дозволяється, коли **START** встановлюється в TRUE, а всі інші - коли змінна **OK** в **DELAY** стане TRUE, тобто коли заданий період закінчиться. Етапам (зверху вниз) присвойте імена **Switch1**, **Green2**, **Switch2**, **Green1**, ну і **Init**, звичайно, збереже своє ім'я. **Switch** повинен включати жовту фазу, в **Green1** TRAFFICSIGNAL1 буде зеленим, в **Green2** TRAFFICSIGNAL2 буде зеленим. Нарешті, змініть адресу повернення Init на **Switch1**. Якщо ви все зробили правильно, діаграма повинна виглядати так (рис. 5.16):

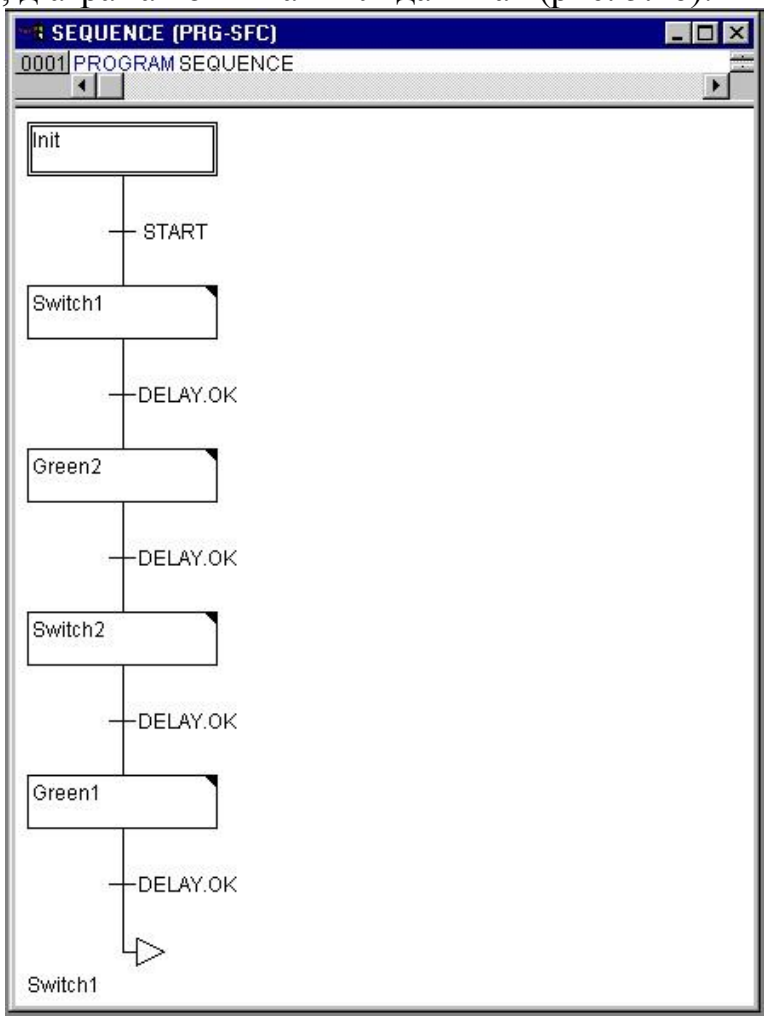

Рисунок 5.16

Тепер ми повинні запрограмувати кроки. Якщо ви зробите подвійний клік ЛКМ на кроці, то відкриється діалогове вікно визначення нової дії. У нашому випадку ми будемо використовувати мову IL (Список Інструкцій) (рис. 5.17).

Рисунок 5.17

Програмування кроків та переходів.

Під час дії кроку Init перевіряємо, активний сигнал включення **START** чи ні. Якщо сигнал не активний, то світлофор вимикається. Цього можна досягти, якщо записати у змінні **TRAFFICSIGNAL1** і **TRAFFICSIGNAL2** число **5.** Ми запишемо в **TRAFFICSIGNAL1 -** 1 а в **TRAFFICSIGNAL2 -** 3, що означає, що на момент увімкнення один світлофор буде в режимі зеленого світла, а інший в режимі червоного (рис. 5.18).

У **Green1** TRAFFICSIGNAL1 буде зеленим (STATUS: = 1), а TRAFFICSIGNAL2 буде червоним (STATUS: = 3), затримка в 5000 мілісекунд (рис. 5.19).

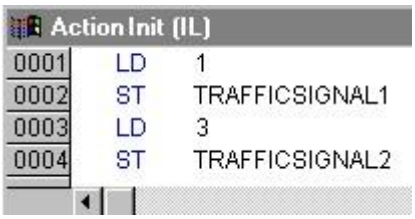

Рисунок 5.18

Рисунок 5.19

Switch1 змінює стан TRAFFICSIGNAL1 на 2 (жовте) і, відповідно, TRAFFICSIGNAL2 на 4 (жовто-червоне). Крім того, тепер встановлюється затримка у 2000 мілісекунд (рис. 5.20).

Green2 включає червоний в TRAFFICSIGNAL1 (STATUS: = 3) і зелений в TRAFFICSIGNAL2 (STATUS: = 1). Затримка встановлюється в 5000 мілісекунд (рис. 5.21).

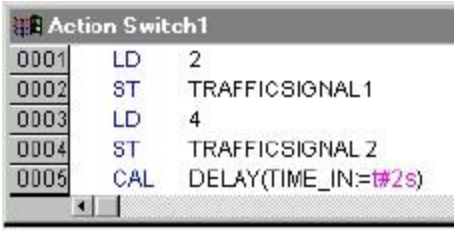

Рисунок 5.20

Рисунок 5.21

У **Switch2** STATUS в TRAFFICSIGNAL1 змінюється на 4 (жовто-червоний), відповідно, TRAFFICSIGNAL2 буде 2 (жовтий). Затримка тепер повинна бути у 2000 мілісекунд (рис. 5.22).

Рисунок 5.22

Загальний алгоритм перемикання ламп світлофора готовий. Тепер розумно буде вимикати наші світлофори на ніч. Для цього ми створимо в програмі лічильник, який після деякого числа циклів TRAFFICSIGNAL зробить відключення пристрою.

Для початку нам потрібна нова змінна **COUNTER** типу INT. Оголосіть її як зазвичай в розділі оголошень SEQUENCE. Тепер виберіть перехід після **Switch1** і вставте ще один крок і перехід. Виберіть результуючий перехід і вставте альтернативну гілку вправо (**Alternative Branch (right)**). Після лівого переходу вставте додатковий крок і перехід. Після нового результуючого переходу вставте віддалений перехід (**Jump**) на Init. Назвіть нові частини так: верхній з двох нових етапів потрібно назвати "**Count**" і нижній "**Off**". Переходи будуть називатися (зверху вниз зліва на право) **EXIT**, **TRUE** і **DELAY.OK**. тепер нові частини повинні виглядати як фрагмент, виділений рамкою (рис. 5.23).

Тепер два нових кроки і переходи необхідно наповнити змістом.

В кроці **Count** виконується тільки одна дія - **COUNTER** збільшується на **1** (рис. 5.24). На переході **EXIT** перевіряється досягнення лічильником заданого значення, наприклад 7 (рис. 5.25):

На етапі **Off** стан обох світлофорів встановлюється в **5** (світлофор вимкнений), **COUNTER** скидається в 0 і встановлюється затримка часу в 10 секунд (рис. 5.26).

Результат.

У нашій гіпотетичній ситуації ніч настає після семи циклів TRAFFICSIGNAL. Світлофори повністю вимикаються до світанку, і процес повторюється знову.

Програмуємо PLC_PRG.

Ми визначили два строго корельовано в часі світлофори в блоці **SEQUENCE**. Тепер повністю закінчимо програму. Для цього необхідно розподілити вхідні і вихідні змінні в блоці **PLC_PRG**. Ми хочемо дати можливість запустити систему вимикачем IN і хочемо забезпечити перемикання всіх шести ламп (2 світлофори по 3 лампи) шляхом передачі "команд переключення" на кожному кроці **SEQUENCE**. Оголосимо тепер відповідні змінні (типу BOOL) для всіх шести виходів та одного входу (кнопки запуску програми), потім створимо програму і співставимо змінні відповідним IEC адресам.

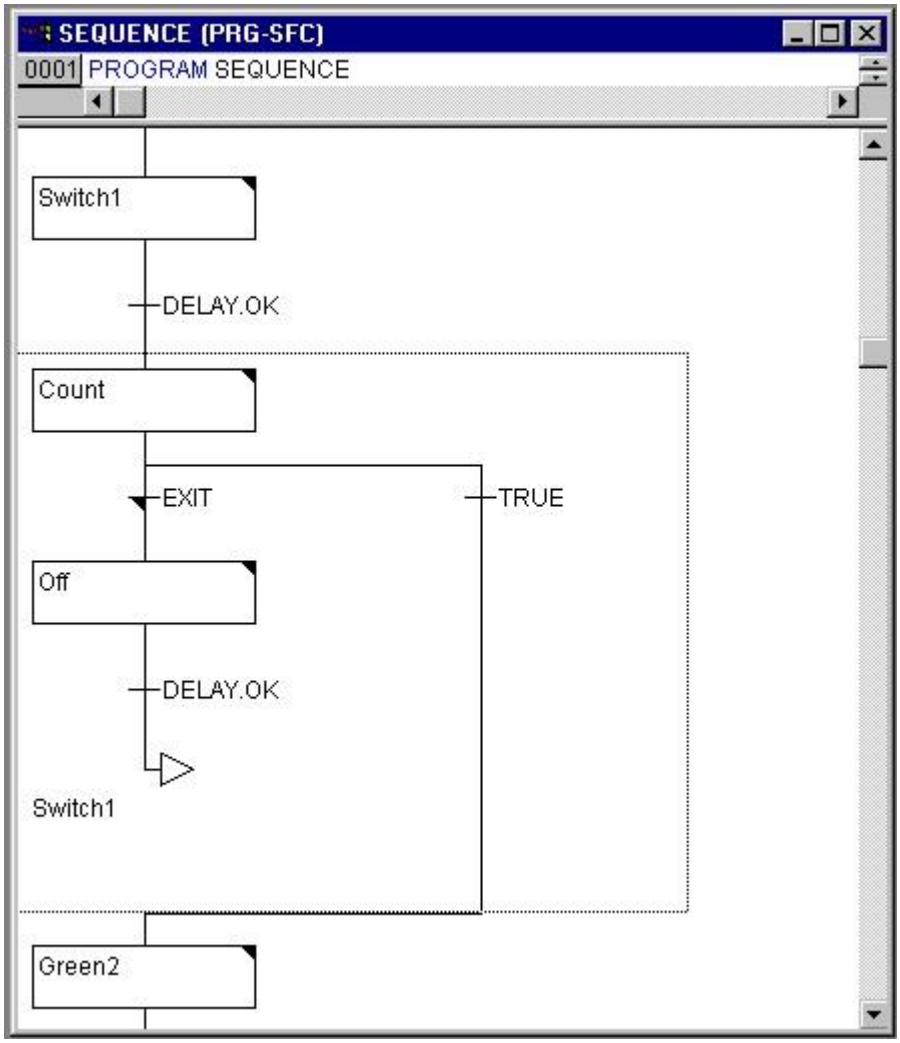

Рисунок 5.23

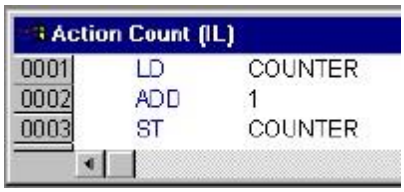

Рисунок 5.24

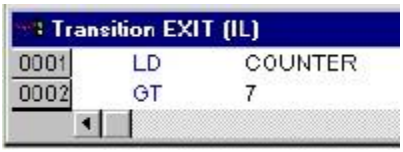

Рисунок 5.25

Наступний крок - це оголошення змінних **LIGHT1** і **LIGHT2** типу TRAFFICSIGNAL в редакторі оголошень (рис. 5.27).

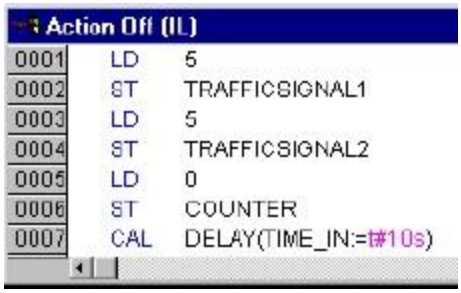

Рисунок 5.26

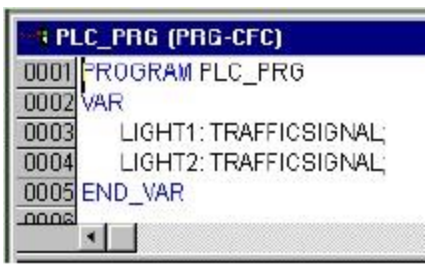

Рисунок 5.27

Для подання шести ламп світлофорів потрібно 6 змінних типу BOOL. Однак ми не будемо оголошувати їх в розділі оголошень блоку **PLC_PRG**, замість цього використовуємо глобальні змінні (Global Variables) із закладки

ресурси (Resources). Двійкова вхідна змінна **IN**, необхідна для установки змінної **START** блоку **SEQUENCE** в TRUE, буде визначена таким же чином.

Виберіть вкладку 'Ресурси' (Resources) і відкрийте список 'Глобальні змінні' (Global Variables). Оголосіть змінну, що буде відповідати за кнопку запуску програми (**IN** типу BOOL) та 6 змінних, що відповідають за лампи світлофорів (червоне, жовте, зелене світло для першого і другого світлофорів) (рис. 5.28):

```
0001 VAR_GLOBAL
0002     IN:BOOL;
0003     L1_GREEN:BOOL;
0004     L1_YELLOW:BOOL;
0005     L1_RED:BOOL;
0006     L2_GREEN:BOOL;
0007     L2_YELLOW:BOOL;
0008     L2_RED:BOOL;
0009 END_VAR
```

Рисунок 5.28

Закінчимо PLC_PRG. Для цього перейдемо у вікно редактора. Ми вибрали редактор Безперервних Функціональних Схем (CFC), і, отже, нам доступна відповідна панель інструментів.

Клікніть ПКМ у вікні редактора і виберіть **Елемент (Box).** Клікніть на тексті AND і напишіть "**SEQUENCE**". Елемент автоматично перетвориться в SEQUENCE з уже певними вхідними та вихідними змінними.

Вставте далі два елементи і назвіть їх **TRAFFICSIGNAL.** TRAFFICSIGNAL - це функціональний блок, і, як зазвичай, Ви отримаєте три червоних знаки питання, які потрібно замінити вже оголошеними локальними змінними **LIGHT1** і **LIGHT2**.

Тепер створіть елемент типу **Вхід (Input)**, який отримає назву **IN** і шість елементів типу **Вихід (Output),** яким потрібно дати такі ж імена як відповідним глобальним змінним, котрі ми вже оголосили: **L1_green**, **L1_yellow**, **L1_red**, **L2_green**, **L2_yellow**, **L2_red**.

Всі елементи програми тепер на місці, і Ви можете з'єднувати входи і виходи. Для цього клацніть на короткій лінії входу / виходу та тягніть її (не відпускаючи клавішу миші) до входу / виходу потрібного елемента.

Нарешті Ваша програма повинна прийняти вигляд, показаний нижче (рис. 5.29)

Тепер наша програма повністю готова, скомпілюйте її як ми робили в попередніх роботах і приступайте до візуалізації.

Візуалізація.

За допомогою візуалізації можна швидко і легко перевірити алгоритм роботи нашого проекту. Зараз ми намалюємо два світлофора і їх вимикач, який дозволить нам включати і вимикати блок управління світлофором.

Для того щоб створити візуалізацію, виберіть вкладку 'візуалізації' (Visualizations) в організаторі об'єктів. Тепер виконайте команду **'Проект' 'Об'єкт - Додати' (Project / Object / Add)**. Введіть будь-яке ім'я для візуалізації, наприклад **Lights**. Коли Ви натиснете кнопку Ok, відкриється вікно, в якому ви будете створювати візуалізацію.

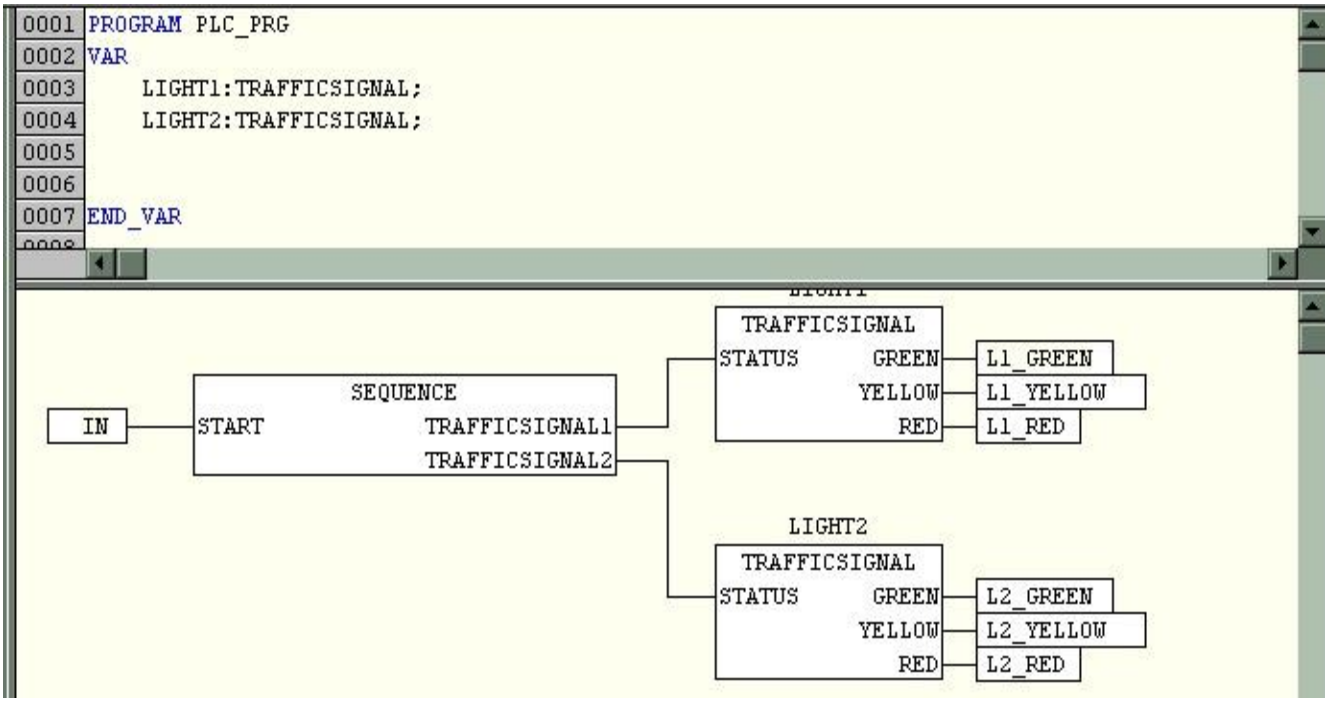

Рисунок 5.29

Для створення візуалізації світлофора виконайте наступні дії:

· Виберіть команду **'Вставка' 'Еліпс' ('Insert' 'Ellipse')** і намалюйте коло з діаметром близько 2 сантиметрів. Для цього клацніть на робочому полі і, утримуючи ЛКМ, розтягніть коло до необхідного розміру.

· Двічі клацніть на колі. З'явиться діалогове вікно для налаштування елемента візуалізації.

· Виберіть категорію **'Змінні' (Variables)** і в полі **'Зміни кольору '(Change color)** введіть ім'я змінної **.L1_red**. Вводити ім'я змінної зручно за допомогою **Асистента Вводу (Input Assistant)** (Клавіша <**F2**>). Глобальна змінна **L1_red** буде керувати кольором намальованого Вами кола.

· Виберіть категорію **'Кольори' (Colors)**. В області **'Кольори' (Color)** натисніть кнопку **'Заливка'(Inside)** і у вікні виберіть будь-який нейтральний колір, наприклад, чорний.

· Натисніть кнопку **'Заливка' (Inside)** в області **'Тривожний колір' (Alarm Color)** і виберіть червоний колір.

Отримане коло буде чорного кольору, коли значення змінної FALSE, і червоного кольору, коли змінна матиме значення TRUE.

Таким чином, ми створили першу лампу першого світлофора. Щоб створити інші кольори світлофора викличте команду **'Редагування' 'Копіювати' ('Edit' 'Copy')** (**Ctrl+C**) і двічі виконайте команду **'Редагування' 'Вставити' ('Edit' 'Paste')** (**Ctrl+V**). Ви отримаєте два нових кола.

Переміщати ці кола можна за допомогою мишки. Розмістіть їх так, щоб вони представляли собою вертикальний ряд в лівій частині вікна редактора. Подвійне клікання по колу призводить до відкриття вікна для налаштування властивостей елемента візуалізації. В полі **'Зміни кольору' (Change Color)** діалогу **'Змінні' (Variables)** вікон налаштування властивостей відповідних кіл введіть наступні змінні:

- для середнього кола: **.L1_yellow**
- для нижнього кола: **.L1_green**

В категорії **'Кольори' (Colors)** в області **'Тривожний колір' (Alarm color)** встановіть колір кіл (жовтий і зелений).

Корпус світлофора. Тепер викличте команду **'Вставка' / 'Прямокутник' ("Insert" "Rectangle")** і вставте прямокутник так, щоб введені раніше кола знаходилися всередині нього. Виберіть колір прямокутника і потім виконайте команду **'Доповнення' 'На задній план' ("Extras" "Send to back")**, яка перемістить його на задній план. Після цього кола знову буде видно.

Активізуйте режим емуляції, виконавши команду **'Онлайн' 'Режим емуляції' - 'Online' "Simulation mode"** (режим емуляції активний, якщо перед пунктом 'Режим емуляції' стоїть галочка). Запустіть програму шляхом виконання команд **'Онлайн' 'Підключитися' ('Online' 'Login')** і **'Онлайн' 'Старт' ('Online' 'Run')** і ви побачите, як будуть змінюватися кольори ламп світлофора.

Другий світлофор. Найпростіший спосіб створити другий світлофор - скопіювати всі елементи першого. Виділіть елементи першого світлофора і скопіюйте їх, виконавши команди **'Редагування' 'Копіювати' ("Edit" "Copy")** і **'Редагування' 'Вставити' ("Edit" "Paste")**. Замініть імена змінних, керуючих кольорами (наприклад, **.L1_red** на **.L2_red**), і другий світлофор буде готовий.

Перемикач ON. Як описано вище, вставте прямокутник, встановіть його колір і введіть змінну **.ON** в поле **'Змін. кольори '(Change Color)** категорії **'Змінні' (Variables)**. В поле **'Рядок' (Content)** категорії **'Текст' (Text)** введіть ім'я "**ON**". Для того щоб змінна ON переключалась при натисканні мишкою на цьому елементі, в поле '**Змінна перемикач**' **(Toggle variable)** категорії **'Введення' (Input)** введіть змінну **.ON**. Створений нами перемикач (кнопка) буде вмикати / вимикати світлофори. Відобразити включений стан можна кольором, як і для світлофора. Впишіть відповідну змінну в поле **'Зміни кольору '(Change Color)**.

Написи в візуалізації. Під світлофорами вставимо два прямокутника. У властивостях елементу в категорії **'Кольори' (Colors)** колір лінії (**frame**) прямокутника задайте білим. В поле **'Рядок' - Contents** (категорія **'Текст' - Text**) введіть назви світлофорів "**Light1**" "**Light2**". Тепер вікно нашого редактора візуалізації виглядає наступним чином (рис. 5.30):

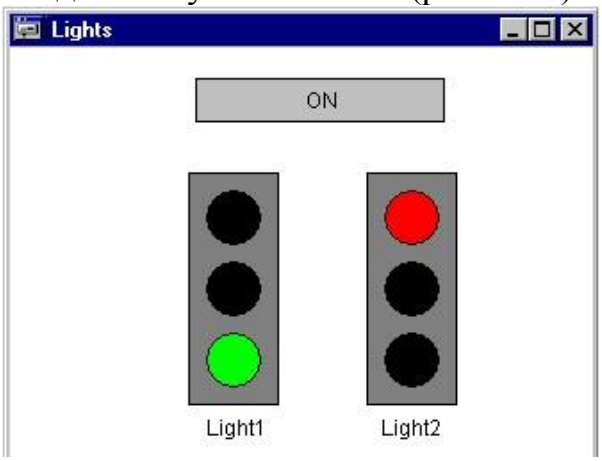

Рисунок 5.30

Запустіть програму на виконання та переконайтеся у правильності роботи проекту.

ПРАКТИЧНА РОБОТА №6

Тема: Вивчення середовища CodeSys для програмування ПЛК ОВЕН. Програмування на мові CFC.

Мета: Розробити програму для керування аварійною сигналізацією перегріву в середовищі CodeSys з використанням таймерів, тригерів та BLINK-алгоритму.

Ціль роботи:

1. Знайомство з основними відомостями про мову програмування CFC.
2. Знайомство із алгоритмами бібліотеки Util.lib.
3. Одержання навичок роботи з тригерами і таймерами.

Короткі теоретичні відомості про мову SFC

Мова SFC (Sequential Function Chart, мова діаграм станів), також відноситься до мов високого рівня, при цьому вона досить значно відрізняється від описаних вище мов. Програма на цій мові являє собою сукупність кроків і умов переходу на наступний крок. Умови або дії під час кроків можуть бути записані на будь-якій мові, як на SFC, так і на будь-якій іншій. Приклад програми на мові SFC показаний на рис. 6.1, на жаль, він не настільки наочний, як в попередніх випадках, тому що дії, що знаходяться всередині кроків не виводяться на екран без виклику. Необхідно відзначити, що для розробки програм на мові SFC повинна бути підключена бібліотека iecsfc.lib.

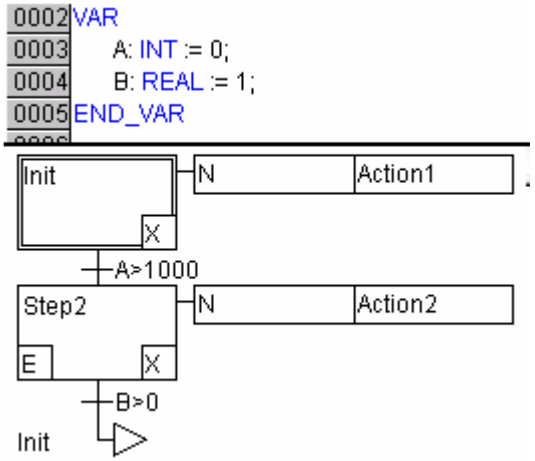

Рисунок 6.1 Приклад програми на мові SFC.

SFC – це графічна мова, яка дозволяє описати хронологічну послідовність різних дій в програмі. Для цього 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 чи ні.

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

Дія

Дія може містити список інструкцій на IL або ST, схеми на FBD або LD, або знову схеми на SFC. При використанні простих кроків дія завжди пов'язується з цим кроком. Для того щоб редактувати дію, необхідно двічі клацнути лівою клавішею мишки на кроці, або виділити крок і вибрати команду меню '**Дополнения**' '**Открыть действие/Переход**' ('Extras' 'Zoom Action / Transition «). Крім основної дії, крок може включати одну вхідну і одну вихідну дію.

Дії МЕК кроків показані в Організаторі Об'єктів, безпосередньо під POU, що їх викликає. Редагування дії запускається подвійним клацанням миші або клавішею <Enter>. Нові дії додаються командою головного меню 'Проект' 'Добавить действие' ('Project' 'Add Action').

Ви можете зіставити одному кроку до 9 дій.

Перехід / умова переходу

Між кроками знаходяться так звані переходи. Умовою переходу може бути логічна змінна або константа, логічна адреса або логічний вираз, описані на будь-якій мові. Умова може включати кілька інструкцій, що утворюють логічний результат, у вигляді ST виразу або на будь-якій іншій мові. Але умова не повинна містити присвоювання, виклик програм і примірників функціональних блоків.

У редакторі SFC умову переходу можна записати безпосередньо біля символу переходу або в окремому вікні редактора для введення умови.

Альтернативна гілка

Дві і більше гілки SFC можуть бути альтернативними. Кожна альтернативна гілка повинна починатися і закінчуватися переходом. Альтернативні гілки можуть містити паралельні гілки та інші альтернативні гілки. Альтернативна гілка починається горизонтальною лінією (початок альтернативи), а закінчується горизонтальною лінією (кінець альтернативи) 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 слідуэ за першим зліва 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 та інші паралельні гілки. Паралельна гілка наноситься подвійною горизонтальною лінією (паралельний початок) 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 слідують паралельні гілки, то активуються перші кроки цих гілок. Ці гілки виконуються паралельно один одному. Крок, який перебуває після паралельних гілок, стає активним лише тоді, коли всі попередні кроки активні і умова переходу істинна.

Завдання для розробки.

Розробити керуючу програму для ПЛК Овен використовуючи мову програмування CFC для керування сигналізацією перегріву. Розробити візуалізацію (мнемосхему), в якій передбачити лампи: «сигнальна лампа перегріву», «увімкнення охолоджувача», та кнопку «скидання аварійної сигналізації». Лампа «сигнальна лампа перегріву» повинна вмикатись через час t_1 після досягнення температури T та працювати в режимі стробоскопа (час при якому лампа світиться t_2 , час при якому лампа не світиться t_3). У візуалізації передбачити можливість ручного задання температури, та табло відображення поточної температури. У програмі використати таймер типу TON, SR-тригер та алгоритм BLINK (табл. 6).

Таблиця 6 Варіанти завдань

№ п/п	Температура T, °C	Час t_1, с	Час t_2, с	Час t_3, с	Нижня границя задання температури	Верхня границя задання температури
1	40	0,5	1	1	20	100
2	50	1,0	0,5	0,5	25	110
3	60	1,5	1	0,5	30	105
4	70	2,0	1,5	1	35	115
5	80	2,5	0,8	0,8	40	120
6	90	3,0	1,2	0,8	15	125
7	100	1,8	1,4	1,2	30	130
8	110	2,2	1,5	1,5	20	135

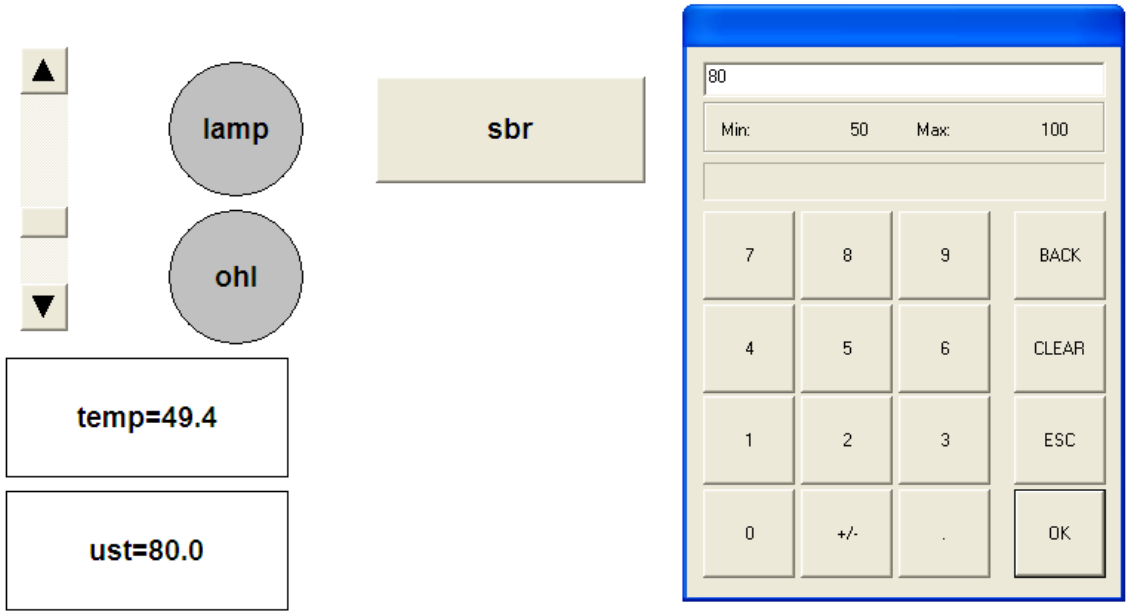

Рисунок 6.2 Загальний вигляд вікна візуалізації проекту

Порядок виконання.

Створіть новий проект обравши пункт верхнього меню **File / New.** Оберіть модель мікроконтролера **ПЛК Овен 110_60М**. Натисніть **ОК** (рис. 6.3).

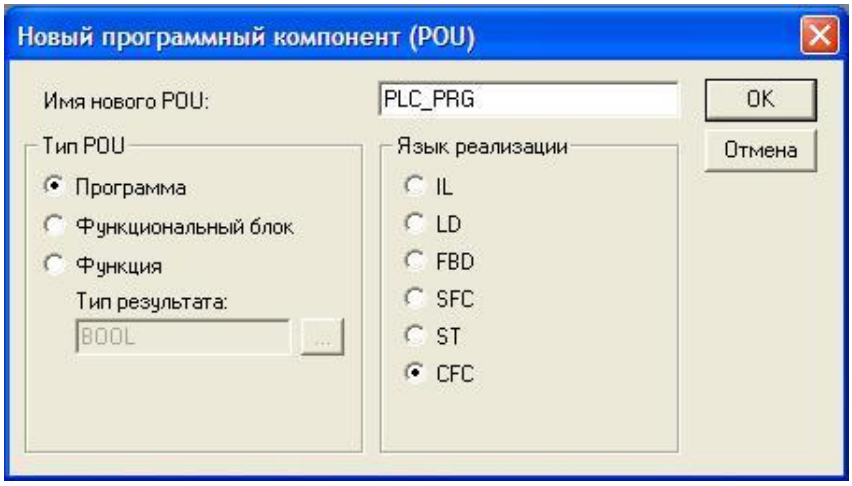

Рисунок 6.3

Нам треба щоб наш охолоджувач вмикався коли температура досягне певного критичного значення. Мовою програмування це можна виразити як: коли значення змінної, що відповідатиме за поточну температуру нашої системи буде перевищувати значення змінної, що відповідатиме за температуру при якій має вмикатися охолоджувач, треба подати сигнал TRUE на змінну BOOL, що буде вмикати охолоджувач (в нашому випадку сигналом його увімкнення буде служити лампочка).

Вставте оператор **Елемент (Box)** із панелі інструментів чи верхнього меню. Замініть напис **AND** на **GT** (оператор порівняння, **більше** «>»). На верхньому вході оператора **GT** оголосіть змінну **Temp** (типу REAL), на нижньому вході – змінну **Ust** (також типу REAL). Тепер наша програма виглядає так (рис. 6.4).

Далі нам треба щоб наш охолоджувач вмикався не відразу коли температура сягне потрібного нам значення для вмикання, а коли температура протримається в цьому значенні хоча б деякий час (наприклад 2с), щоб уникнути випадкового вмикання (наприклад при скачкоподібному зростанні температури внаслідок парової пробки). Для таких цілей нам підійде таймер затримки включення (типу TON). Вставте **Елемент (Box)**, і замініть напис **AND** на **TON.** Введіть назву таймера замість трьох знаків питання, наприклад **Timer1.** Підключіть вихід оператора **GT** на вхід **IN** таймера **TON.** На вході **PT** таймера введіть константу **T#2s** (затримка увімкнення 2 сек.) (рис. 6.5).

Рисунок 6.4

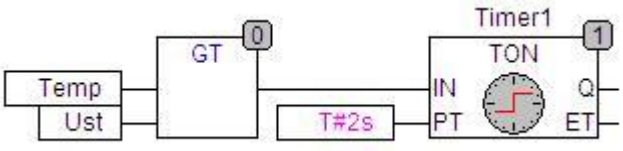

Рисунок 6.5

Далі нам треба щоб оператор мав можливість вимкнути лампочку охолоджувача, коли температура знизиться. Для цього використаємо SR-тригер. Вставте **Елемент (Box)**, і замініть напис **AND** на **SR.** Введіть назву

тригера, наприклад **Sr1.** Підключіть вихід **Q** таймера на вхід **SET** тригера. Оголосіть зінну **Sbr** типу BOOL на вході **RESET** тригера. Оголосіть зінну **Ohl** типу BOOL на виході **Q** тригера. Тепер наша програма має наступний вигляд (рис. 6.6):

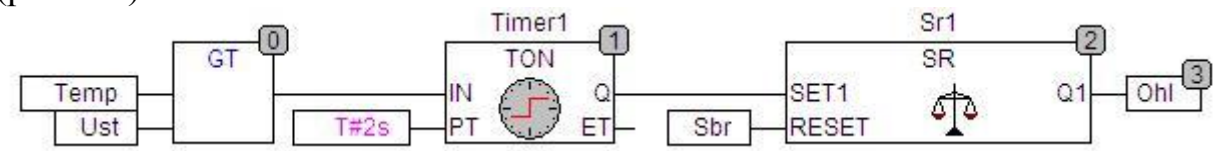

Рисунок 6.6

Наступним кроком нам треба запрограмувати сигнальну лампу перегріву таким чином, щоб вона працювала в «мигаючому» режимі роботи. Для цього звичайно ж можна застосувати таймер, але ми використаємо готовий алгоритм **BLINK**. Це нестандартний алгоритм, він знаходиться в бібліотеці утиліт **Util.lib**. Тому щоб його вставити в програму, спочатку слід підключити цю бібліотеку.

Перейдіть в закладку **Ресурси / Менеджер бібліотек (Resources / Library Manager)**. Клікніть ПКМ у вікні зі списком вже підключених бібліотек, і виберіть пункт контекстного меню **Добавити бібліотеку (Additional library)**. У діалоговому вікні виберіть бібліотеку **Util.lib** та натисніть **Відкрити (Open)** (рис. 6.7).

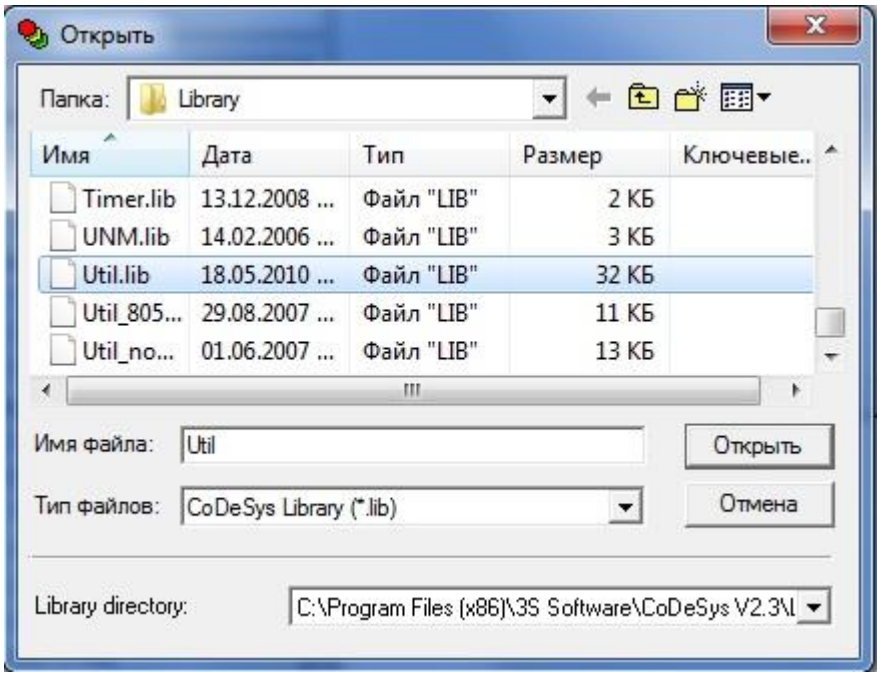

Рисунок 6.7

Тепер бібліотека **Util.lib** з’явилася у списку підключених бібліотек (рис. 6.8).

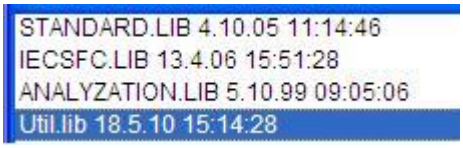

Рисунок 6.8

Таким чином бібліотека підключена, і тепер, ми можемо повернутись у PLC_PRG. Вставте **Елемент (Box)**, і замініть напис **AND** на **BLINK.** Замініть

три знаки питання на ім'я змінної (наприклад **Blink1**). Тепер нам необхідна змінна типу BOOL, яка вмикатиме наш алгоритм **BLINK.** Оголосимо цю змінну на вході **ENABLE** алгоритму **BLINK**. Дамо їй назву, наприклад **Alarm.** Вона вмикатиме наш алгоритм **BLINK**. Вмикатися наша сигнальну лампу перегріву має разом із охолоджувачем. То ж пропишемо її також на виході **Q** тригера (рис. 6.9).

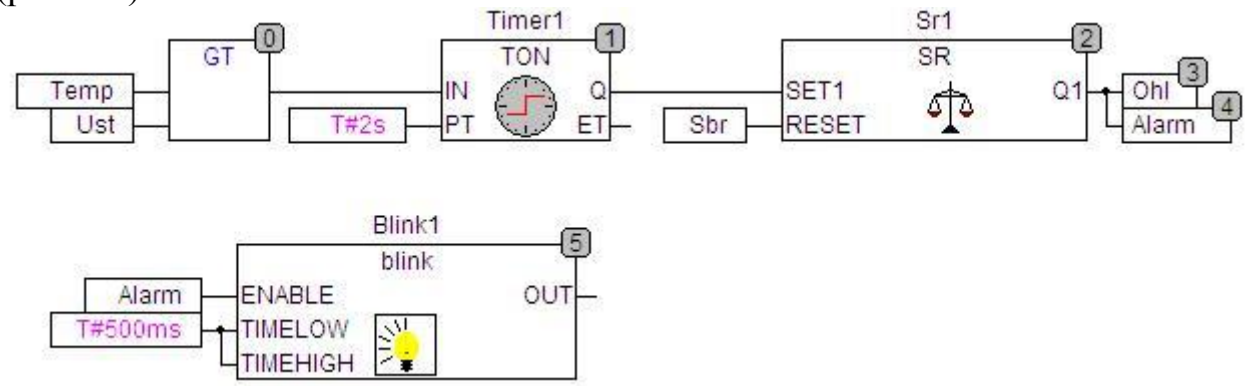

Рисунок 6.9

Задайте уставку часу 0,5 сек на входи TIMELOW і TIMEHIGH алгоритму **BLINK.** Тепер нам потрібна змінна типу BOOL, яка власне відповідатиме за нашу сигнальну лампу перегріву. Оголосимо її з ім'ям, наприклад **Lamp**. Якщо ми підключимо нашу лампу на вихід **OUT** алгоритму **BLINK,** то в нас може виникнути наступна ситуація: якщо на вхід **ENABLE** алгоритму **BLINK** прийде сигнал **FALSE**, коли лампа увімкнена, то алгоритм припинить роботу, а лампа залишиться увімкненою. Нас така ситуація не влаштовує. Нам треба щоб лампа не тільки перестала «мигати», а ще й гасла. Тому ми прив'яжемо її увімнення із увімкненням алгоритму **BLINK**. Таким чином при вимкненні алгоритму буде вимикатися і лампа. Вставте **Елемент (Box).** Підключіть вихід **OUT** алгоритму **BLINK** на один із входів **AND**. Інший вхід оператора **AND** під'єднайте на вхід алгоритму **BLINK.** На вихід **AND** подайте змінну **Lamp.**

Розставте порядок виконання. Клікніть ПКМ на полі програми та оберіть у контекстному меню пункт **Порядок / У відповідності із потоком данних** (**Order / Order everything according to data flow**). Остаточний вигляд нашої програми (рис. 6.10):

Компіляція проекту.

Відкомпілюйте проект повністю командою верхнього меню «**Проэкт / Компилировать все**» (**Project / Rebuild all**), або скористайтеся клавішею «**F11**». Якщо ви все виконали вірно, то в нижній частині вікна повідомлень має з'явитись надпис «0 errors». В іншому випадку необхідно виправити допущені помилки. В цьому допоможуть повідомлення про помилки.

Візуалізація проекту.

Добавте елемент візуалізації в третій закладці лівого вікна CoDeSys (**Vizualization / Add object**). Перейдіть на сторінку візуалізації. Присвойте новому об'єкту ім'я. В кінці роботи вікно візуалізації має виглядати наступним чином (рис. 6.11):

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     Temp: REAL;
0004     Alarm: BOOL;      (*Аварійний сигнал*)
0005     Ust: REAL := 80;  (*установка*)
0006     Sr1: SR;
0007     Sbr: BOOL;
0008     Ohl: BOOL;
0009     Lamp: BOOL;
0010     Timer1: TON;
0011     Blink1: BLINK;
0012 END_VAR
```

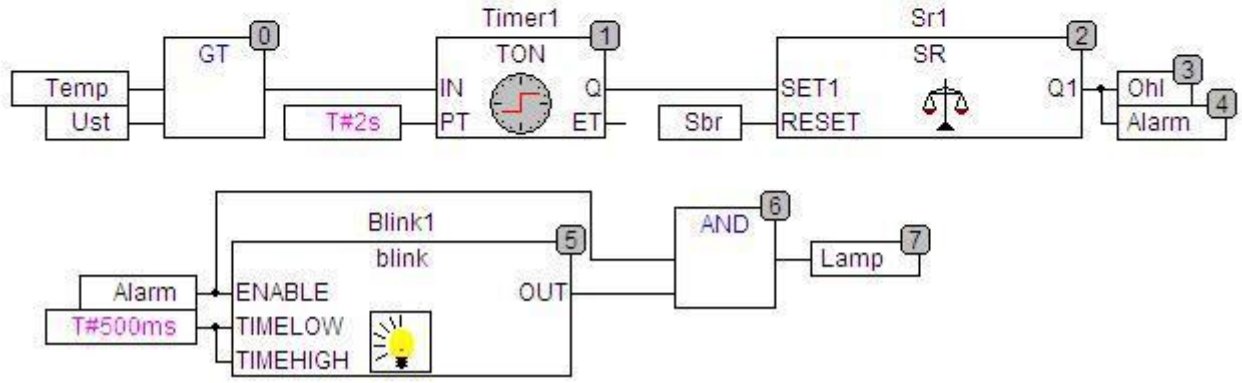

Рисунок 6.10

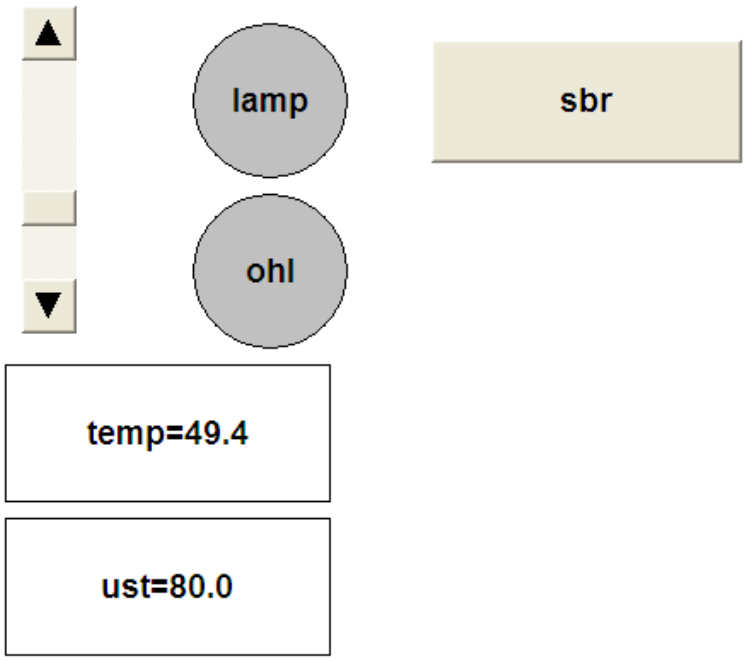

Рисунок 6.11

Для створення візуалізації виконайте наступні дії:

Виберіть команду **'Вставка' 'Еліпс' ('Insert' 'Ellipse')** і намалюйте коло з діаметром близько 2 сантиметрів. Двічі клікніть ЛКМ на колі. З'явиться діалогове вікно для налаштування елемента візуалізації. Виберіть категорію **'Змінні' (Variables)** і в полі **'Зміни кольору '(Change color)** введіть ім'я змінної **«.Lamp»**. Виберіть категорію **'Кольори' (Colors)**. В області **'Кольори' (Color)** натисніть кнопку **'Заливка'(Inside)** і у вікні виберіть будь-який нейтральний

колір, наприклад, чорний. Натисніть кнопку **'Заливка' (Inside)** в області **'Тривожний колір' (Alarm Color)** і виберіть червоний колір. В поле **'Рядок' (Content)** категорії **'Текст' (Text)** введіть ім'я "**Lamp**". Таким чином, ми створили першу лампу. Щоб створити іншу викличте команду **'Редагування' 'Копіювати' ('Edit' 'Copy') (Ctrl+C)** і команду **'Редагування' 'Вставити' ('Edit' 'Paste') (Ctrl+V)**.

Подвійне клікання по колу призводить до відкриття вікна для налаштування властивостей елемента візуалізації. В полі **'Зміни кольору' (Change Color)** діалогу **'Змінні' (Variables)** вікон налаштування властивостей введіть ім'я змінної «**.Ohl».**

Перемикач Sbr.

Вставте кнопку (**Button**) як зображено у завданні. В поле **'Рядок' (Content)** категорії **'Текст' (Text)** введіть ім'я "**Sbr**". В поле **'Змінна перемикач' (Toggle variable)** категорії **'Введення' (Input)** введіть змінну **.Sbr**. Створений нами перемикач (кнопка) буде вимикати лампи після зниження температури.

Повзунок.

Виберіть команду **Вставити / Повзунок** (**Scrollbar**). Розтягніть повзунок аналогічно до побудови прямокутника. Двічі клікніть для відкриття вікна конфігурації елемента. В категорії змінні введіть мінімальне та максимальне значення температури згідно вашого завдання. В полі **Повзунок** введіть змінну «**.Temp**». Натисніть **ОК**.

Відображення температури.

Побудуйте 2 прямокутники. Перший прямокутник (верхній) буде відображати поточне значення температури. Другий (нижній) буде відображати задаючу температуру увімкнення охолоджувача, та змінювати її значення. Відредагуйте перший прямокутник наступним чином:

- В полі **Виведення тексту' (Text display)** діалогу **'Змінні' (Variables)** введіть ім'я змінної «**.Temp».**
- В поле **'Рядок' (Content)** категорії **'Текст' (Text)** введіть "temp=%3.1f" (рис. 6.12).

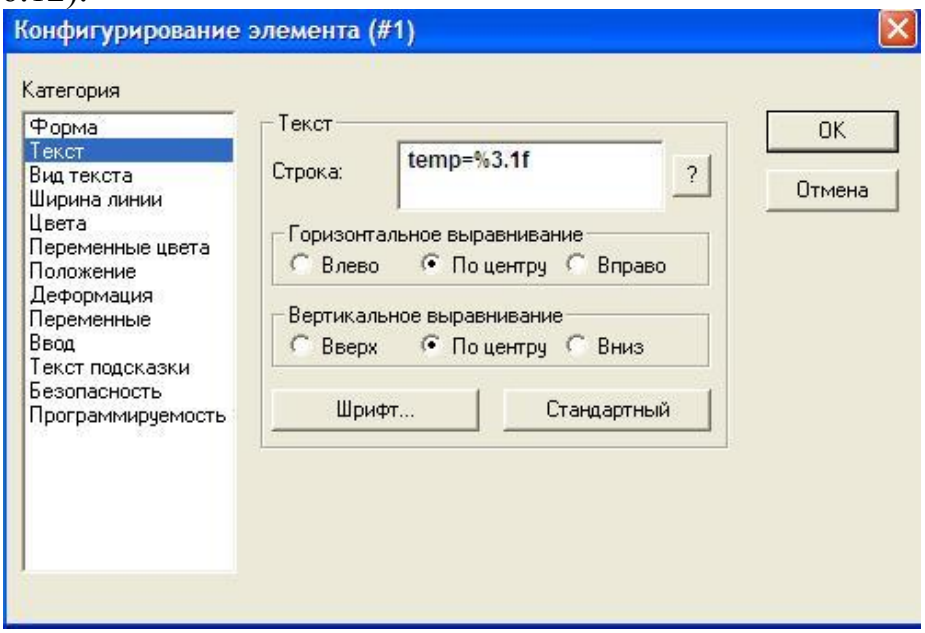

Рисунок 6.12

Відредагуйте другий прямокутник наступним чином:

- В полі ‘**Виведення тексту' (Text display)** діалогу **'Змінні' (Variables)** введіть ім'я змінної «**.Ust».**
- Поставте галочку в поле **'Ввід у змінну «Виведення тексту»' (Text input of variable «Text display»)** категорії **'Введення' (Input)** та введіть змінну **.Sbr**. У випадаючому списку виберіть ‘**Числова панель**’ (**Numpad**) та задайте мінімальне і максимальне значення (рис. 6.13).

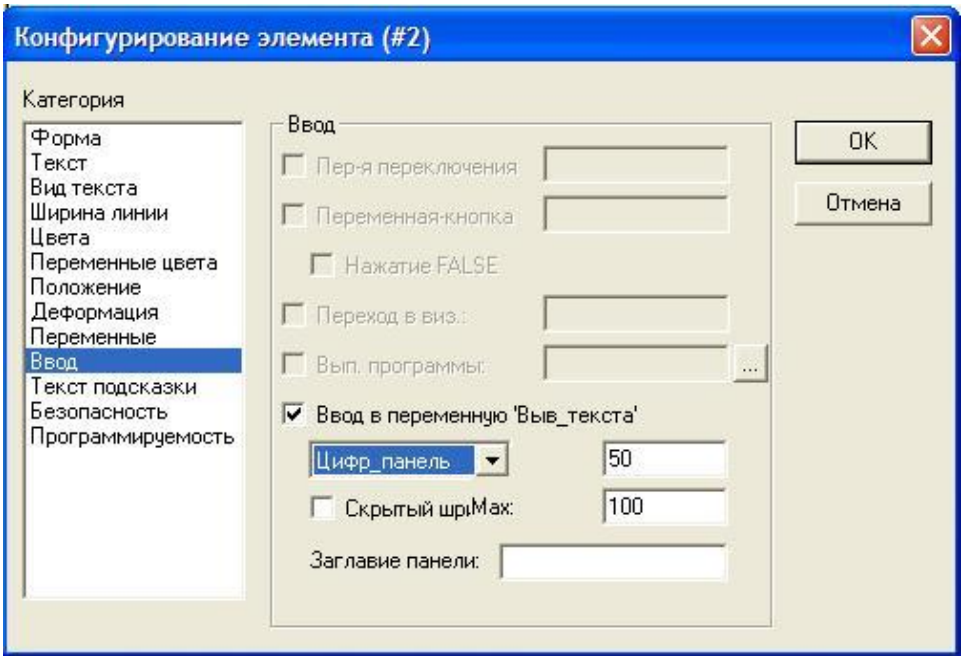

Рисунок 6.13

- В поле **'Рядок' (Content)** категорії **'Текст' (Text)** введіть "**ust=%3.1f**". (рис. 6.14):

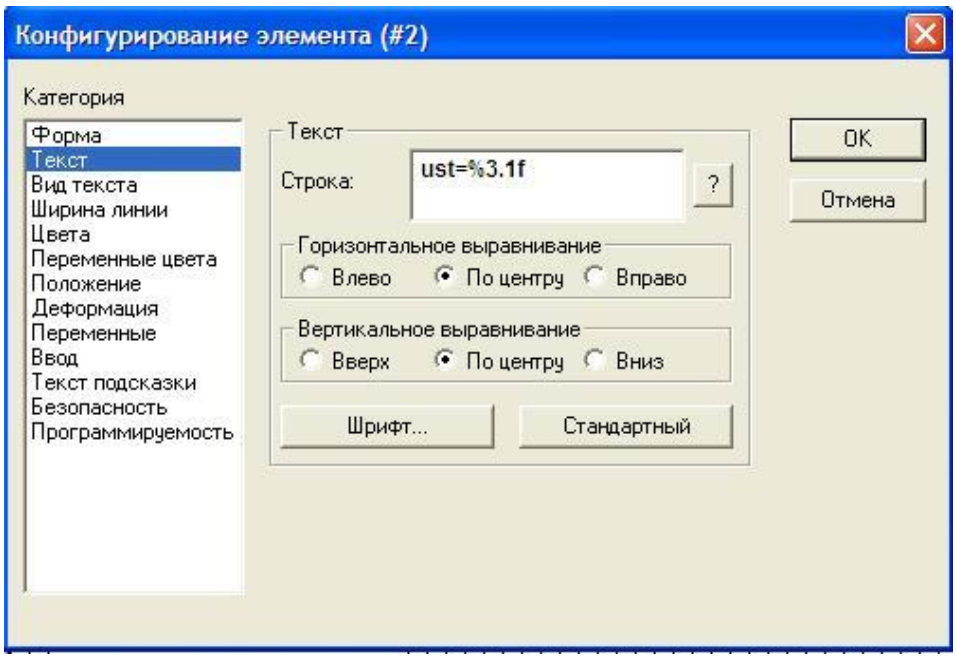

Рисунок 6.14

Працює наша програма наступним чином:

Коли користувач повзунком піднімає температуру її поточне значення відображає верхнє прямокутне табло. Коли ця температура перевищує величину, яка відображена в нижньому прямокутному табло, через деякий заданий час загорається в «мигаючому» режимі сигнальна лампа перегріву **Lamp** та вмикається охолоджувач, про що свідчить світіння лампи **Ohl.** Порогове значення температури ввімкнення охолоджувача в нижньому табло можна змінювати. При цьому відкривається цифрове табло із вказаними межами задання порогового значення температури.

Після зниження температури оператор може вимкнути лампи натиснувши кнопку **Sbr**.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання лабораторних робіт на тему “ Програмування програмованих логічних контролерів фірми Шнайдер Електрик” з дисципліни “Програмне забезпечення комп’ютерно-інтегрованих технологій” /Уклад. Микитишин А.Г., Митник М.М. – Тернопіль: ТДТУ, 2001. – 73
2. Programable controllers. Programming languages. Technical committee no. 65: industrial-process measurement and controlsub-committee 65b: devices. Working group 7: programmable controllers voting draft - iec 61131-3, 2nd ed. Programmable controllers - programming languages.
3. TLX DR PL7 12E. PL7 Micro/Junior. Reference Manual (Version V1.2).
4. Сайт фірми 3S-Smart Software Solutions GmbH www.3s-software.com.
5. Сайт фірми НПФ Пролог www.codesys.ru
6. Сайт фірми ОВЕН www.owen.ru.
7. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. – М.: СОЛОН-ПРЕСС, 2007. – 256 с.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчально-методична література

Методичні вказівки до виконання практичних робіт

з дисципліни

«Автоматизовані системи керування технологічними процесами»

напряму підготовки 6.050202 «Автоматизація та комп'ютерно-інтегровані технології»

Редагування *Є.І. Гриценко*
Комп'ютерне макетування та верстка *А.П. Катрич*

Формат 60х90/16. Обл. вид. арк. 0,77. Тираж 10 прим. Зам. № 2765.

Видавництво Тернопільського національного
технічного університету імені Івана Пулюя.
46001, м. Тернопіль, вул. Руська, 56.
Свідоцтво суб'єкта видавничої справи ДК № 4226 від 08.12.11.