МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

# ТЕХНІЧНІ ЗАСОБИ АВТОМАТИЗАЦІЇ

## ЛАБОРАТОРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньо-професійною програмою
«Комп'ютерно-інтегровані системи та технології в приладобудуванні»
спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2022

Укладачі: *Тимчик Г. С.*, доктор технічних наук, професор
*Антонюк В. С.*, доктор технічних наук, професор
*Здоренко В. Г.*, доктор технічних наук, професор
*Защепкіна Н. М.*, доктор технічних наук, професор
*Лісовець С. М.*, кандидат технічних наук, доцент
*Клочко Т. Р.*, кандидат технічних наук, доцент

Рецензенти: *Мураховський С. А.*, кандидат технічних наук, доцент кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем НТУУ «КПІ ім. Ігоря Сікорського»
*Ківа І. Л.*, кандидат технічних наук, доцент кафедри автоматизованого управління технологічними процесами Таврійського національного університету імені В.І. Вернадського

Відповідальний редактор: *Філіппова М. В.*, кандидат технічних наук, доцент

*Гриф надано Методичною радою КПІ імені Ігоря Сікорського*
*(протокол № 6 від 24.06.2022 р.)*
*за поданням Вченої ради Приладобудівного факультету*
*(протокол № 6/22 від 20.06.2022 р.)*

Технічні засоби автоматизації. Лабораторний практикум [Електронний ресурс] : навчальний посібник для здобувачів ступеня бакалавра за освітньо-професійною програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні» спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / КПІ ім. Ігоря Сікорського ; уклад.: Г. С. Тимчик, В. С. Антонюк, В. Г. Здоренко, Н. М. Защепкіна, С. М. Лісовець, Т. Р. Клочко. – Електронні текстові дані (1 файл: 1,17 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022. – 174 с. – Назва з екрана.

В учбовому лабораторному практикумі викладені загальні принципи створення програмного коду для програмованих логічних контролерів (ПЛК), які широко застосовуються як для побудови промислових виробничих ліній систем автоматизації, так і для різноманітних гнучких систем керування побутового призначення. Наводяться основи із створення коду найпростіших програм для сучасних ПЛК на різних мовах програмування, що використовуються сьогодні для більшості автоматизованих систем промислового призначення. Приводяться завдання для ознайомлення з роботою створення різних програм в середовищі програмування CoDeSys фірми 3S майбутніх фахівців з автоматизації та комп'ютерно-інтегрованих систем. Посібник призначений для студентів вищих закладів освіти, а також інженерно-технічних працівників у галузі автоматизації та приладобудування.

Реєстр. № НП 21/22-683. Обсяг 7,37 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua

Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

# ДЕЯКІ ТЕРМІНИ І ВИЗНАЧЕННЯ, ЯКІ ВІДНОСЯТЬСЯ ДО АВТОМАТИЗОВАНИХ СИСТЕМ

**Метрологічне забезпечення автоматизованої системи керування технологічними процесами** – встановлення і застосування наукових і організаційних основ, технічних і програмних засобів, правил, норм з метою забезпечення єдності і заданої точності вимірювань та заданої точності керувальних дій на технологічний об'єкт керування, які здійснюються автоматизованою системою керування технологічними процесами.

**Автоматизована система** – організаційно-технічна система, яка складається із засобів автоматизації певного виду (або кількох видів) діяльності людей і персоналу, що здійснює цю діяльність.

**Автоматизована система керування технологічними процесами** – автоматизована система, яка призначена для вироблення і реалізації керувальної дії на технологічний об'єкт керування згідно з прийнятими критеріями керування.

**Вимірювальний канал автоматизованої системи керування технологічними процесами** – функціонально об'єднана частина автоматизованої системи керування технологічними процесами, призначена для автоматизованого створення інформативного сигналу про властивості технологічного об'єкта керування, перетворення його в інформаційний сигнал вимірюваної фізичної величини і подання його у вигляді натурального числа і (чи) цифрового коду.

**Обчислювальний канал автоматизованої системи керування технологічними процесами** – функціонально об'єднана частина автоматизованої системи керування технологічними процесами, яка використовується для обчислень під час виконання функцій сумісних і сукупних вимірювань, а також визначення параметрів технологічного об'єкта керування.

**Канал керування автоматизованої системи керування технологічними процесами** – функціонально об'єднана частина автоматизованої системи керування технологічними процесами, яка призначена для формування сигналу технологічної уставки, порівняння з ним інформаційного сигналу вимірюваної фізичної величини, вибору керувальних дій за результатами порівняння і їх реалізації шляхом формування енергетичних сигналів.

**Вимірювальний компонент** – функціонально об'єднана частина вимірювального каналу, яка здійснює одну або декілька вимірювальних операцій.

**Обчислювальний компонент** – сукупність програм (комплексів програм) на носіях даних і обчислювальних засобів, які входять до складу автоматизованої системи керування технологічними процесами, що призначені для виконання обчислювальних операцій під час вимірювань.

**Сигнал керувальної дії** – енергетичний сигнал, який формується на виході каналу керування з метою змінення параметрів технологічного об'єкту керування або їх підтримання на заданому рівні.

**Похибка керувальної дії** – різниця між номінальним і дійсним значеннями параметра сигналу керувальної дії.

**ЗМІСТ**

# ВСТУП

Метою виконання лабораторних робіт є вивчення архітектури програмованих логічних контролерів ПЛК100, ПЛК150 і ПЛК154, які випускаються ТОВ "ВО ОВЕН" (м. Харків, Україна). Програмовані логічні контролери ПЛК100, ПЛК150 і ПЛК154 призначені для малих автоматизованих систем (наприклад, для порівняння, програмовані логічні контролери ТОВ "ВО ОВЕН" ПЛК63 і ПЛК73 призначені для локальних автоматизованих систем, ПЛК110 і ПЛК160 – для середніх автоматизованих систем, ПЛК304 і ПЛК323 – для здійснення комунікаційного обміну). ПЛК100, ПЛК150 і ПЛК154 займають "середнє" положення між програмованими логічними контролерами ТОВ "ВО ОВЕН", і одночасно мають практично всі функції, які властиві як більш молодшим, так і більш старшим моделям. Таким чином, вивчення роботи ПЛК100, ПЛК150 і ПЛК154 дозволяє зрозуміти роботу будь-якого програмованого логічного контролера виробництва ТОВ "ВО ОВЕН".

Виконання лабораторних робіт вимагає роботи в середовищі програмування CoDeSys, яке створено фірмою 3S (Smart Software Solutions) з метою розробки програмного забезпечення для програмованих логічних контролерів різних виробників, в тому числі і програмованих логічних контролерів виробництва ТОВ "ВО ОВЕН".

В кожній лабораторній роботі розглядається окрема апаратна і/або програмна складова ПЛК150 і CoDeSys, які є характерними для типової автоматизованої системи. Таким чином, окреме вивчення роботи таких складових дозволяє зрозуміти роботу типової автоматизованої системи в цілому.

Зокрема, вивчення роботи ПЛК150 дозволяє зрозуміти принципи уведення і виведення аналогових і дискретних сигналів та принципи обміну даними за допомогою комунікаційних інтерфейсів, середовища програмування CoDeSys – принципи програмування програмованих логічних контролерів.

Припускається, що налагодження прикладного програмного забезпечення ПЛК150 спочатку виконуватиметься в програмному середовищі CoDeSys в режимі Simulation (наприклад, в цьому режимі можна виконувати програмний код покроково, продивлятися і змінювати значення змінних, виявляти критичні ділянки прикладного програмного забезпечення тощо).

Після налагодження прикладного програмного забезпечення припускається програмування ПЛК150 безпосередньо з програмного середовища CoDeSys. При цьому передбачається застосування комунікаційного інтерфейсу Ethernet і маршрутизатора TP-LINK 740n (або аналогічного).

Виконання лабораторних робіт також вимагає певного додаткового обладнання. До такого обладнання відносяться термоелектричний перетворювач типу ТХК(L) або ТХА(K), термометр опору типу ТСМ або ТСП, джерело уніфікованих електричних сигналів напруги і струму, контактний перетворювач, аналоговий вольтметр (цифровий вольтметр, мультиметр), аналоговий амперметр (цифровий амперметр, мультиметр) тощо.

# ЛАБОРАТОРНА РОБОТА №1. ПРОГРАМОВАНИЙ ЛОГІЧНИЙ КОНТРОЛЕР ПЛК150 ВИРОБНИЦТВА ТОВ “ВО ОВЕН”

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення програмованого логічного контролера ПЛК150 виробництва ТОВ “ВО ОВЕН”.

## 2. Порядок виконання лабораторної роботи

1. Ознайомитися з характеристиками ПЛК150.

В протокол занести (можна скорочено):

- основні технічні характеристики ПЛК150;
- характеристики ресурсів ПЛК150;
- характеристики аналогових входів ПЛК150;
- характеристики аналогових виходів ПЛК150;
- характеристики дискретних входів ПЛК150;
- характеристики дискретних виходів ПЛК150;
- характеристики інтерфейсів ПЛК150;
- характеристики програмування ПЛК150;
- характеристики дискретних вхідних сигналів ПЛК150;
- характеристики вбудованих цифро-аналогових перетворювачів.

2. З’ясувати, як до ПЛК150 можна підключити:

- вхідні аналогові сигнали;
- вихідні аналогові сигнали;
- вхідні дискретні сигнали;
- вихідні дискретні сигнали.

В протокол занести: схеми електричні підключення до ПЛК150 вхідних і вихідних аналогових і дискретних сигналів.

4. З’ясувати, як ПЛК150 можна підключити до електричної мережі.

В протокол занести: схему електричну підключення ПЛК150 до електричної мережі.

5. З’ясувати, як в ПЛК150 застосовується комунікаційний інтерфейс Ethernet, які комунікаційні протоколи від підтримує.

В протокол занести: схему електричну підключення до ПЛК150 по комунікаційному інтерфейсу Ethernet.

6. З’ясувати, як в ПЛК150 застосовується комунікаційний інтерфейс RS-485, які комунікаційні протоколи від підтримує.

В протокол занести: схему електричну підключення до ПЛК150 по комунікаційному інтерфейсу RS-485.

7. З’ясувати, як в ПЛК150 застосовується комунікаційний інтерфейс Debug RS-232, які комунікаційні протоколи від підтримує.

В протокол занести: схему електричну підключення до ПЛК150 по комунікаційному інтерфейсу Debug RS-232.

8. З’ясувати, для чого призначена кнопка “Старт/Стоп”.

В протокол занести: призначення кнопки “Старт/Стоп”.

9. З’ясувати, для чого призначена кнопка “Сброс”.

В протокол занести: призначення кнопки “Сброс”.

10. З’ясувати, для чого призначені індикатори “Питание”, “Работа” і “Связь”.

В протокол занести: призначення індикаторів “Питание”, “Работа” і “Связь”.

11. З’ясувати, для чого призначені акумулятор і годинник реального часу ПЛК150.

В протокол занести: параметри акумулятора і годинника реального часу ПЛК150.

10. З’ясувати, яку MAC-адресу має ПЛК150.

В протокол занести: MAC-адресу ПЛК150.

11. З’ясувати, яку IP-адресу має ПЛК150.

В протокол занести: IP-адресу ПЛК150.

12. З’ясувати призначення Target-файлів і як виконується їх інсталяція.

В протокол занести: порядок інсталяції Target-файлів.

### 3. Питання для самоперевірки

1. Яке основне призначення ПЛК100, ПЛК110, ПЛК150, ПЛК154?
2. Які типи первинних вимірювальних перетворювачів підтримують ПЛК100, ПЛК150, ПЛК154?
3. З якими типами уніфікованих аналогових вхідних сигналів можуть працювати ПЛК100, ПЛК150, ПЛК154?
4. Які типи уніфікованих аналогових вихідних сигналів можуть формувати ПЛК100, ПЛК150, ПЛК154?
5. Як можна сформувати сигнали лог. 0 і лог. 1 на дискретних входах ПЛК100, ПЛК150, ПЛК154?
6. Як формуються вихідні дискретні сигнали ПЛК100, ПЛК150, ПЛК154?
7. Які комунікаційні інтерфейси мають ПЛК100, ПЛК150, ПЛК154?
8. Що таке Target-файли, як вони використовуються середовищем програмування CoDeSys фірми 3S (Smart Software Solutions)?
9. Які основні апаратні і програмні технічні характеристики (процесора, оперативної пам’яті, енергонезалежної пам’яті, Retain-пам’яті, часу виконання одного циклу, конструктивного виконання тощо) мають ПЛК100, ПЛК150, ПЛК154?
10. Як встановлюється зв’язок з ПЛК100, ПЛК150, ПЛК154 з метою їх програмування за допомогою середовища програмування CoDeSys фірми 3S (Smart Software Solutions)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №2.
# СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ CODESYS ФІРМИ 3S (SMART SOFTWARE SOLUTIONS)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення середовища програмування CoDeSys фірми 3S (Smart Software Solutions).

## 2. Порядок виконання лабораторної роботи

1. Ознайомитися з характеристиками середовища програмування CoDeSys.

В протокол занести (можна скорочено):

- призначення середовища програмування CoDeSys;
- склад, об’єм і уміст документації по середовищу програмування CoDeSys;
- компоненти проекту в CoDeSys.

2. Відкрити приклад проекту в середовищі програмування CoDeSys, наприклад, файл example.pro (поставляється разом з середовищем програмування CoDeSys). Виконати компіляцію проекту.

В протокол занести:

- файли, з яких складається проект;
- каталог, в якому знаходяться файли проекту;
- кількість і тип помилок при виконанні компіляції (якщо вони є);
- кількість і тип попереджень при виконанні компіляції (якщо вони є);
- кількість індексів компонентів організації програм;
- розмір даних, які використовуються;
- розмір RETAIN-даних, які використовуються.

3. З’ясувати, які компоненти організації програм використовуються в проекті.

В протокол занести:

- опис функцій, які використовуються в проекті;
- опис функціональних блоків, які використовуються в проекті;
- опис програм, які використовуються в проекті;
- опис дій, які використовуються в проекті.

4. З’ясувати, які *Глобальні змінні* використовуються в проекті.

В протокол занести: опис *Глобальних змінних*.

5. З’ясувати, які параметри *Конфігурації тривог* використовуються в проекті.

В протокол занести: опис *Конфігурації тривог*.

6. З’ясувати, які параметри *Менеджера бібліотек* використовуються в проекті.

В протокол занести: опис *Менеджера бібліотек*.

7. З’ясувати, які параметри *Бортжурналу* використовуються в проекті.

В протокол занести: опис *Бортжурналу*.

8. З’ясувати, які параметри *Конфігурації ПЛК* використовуються в проекті.

В протокол занести: опис *Конфігурації ПЛК*.

9. З’ясувати, які параметри *Менеджера перегляду* використовуються в проекті.

В протокол занести: опис *Менеджера перегляду*.

10. З’ясувати, які параметри *Конфігурації задач* використовуються в проекті.

В протокол занести: опис *Конфігурації задач*.

11. З’ясувати, які параметри *Налагодження цільової платформи* використовуються в проекті.

В протокол занести: опис *Налагодження цільової платформи*.

12. З’ясувати, які параметри *Робочої області* використовуються в проекті.

В протокол занести: опис *Робочої області*.

### 3. Питання для самоперевірки

1. Що таке проект в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions), які програмні складові входять до складу проекту?
2. Що таке функції, як функції використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
3. Що таке функціональні блоки, як функціональні блоки використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
4. Що таке програми, як програми використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
5. Що таке програма PLC_PRG, як вона використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
6. Що таке дії, як вони використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
7. Що таке ресурси, як вони використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
8. Що таке бібліотеки, як вони використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
9. Що таке стандартні типи даних і типи даних, які “визначаються користувачем” (структури, перелічення, посилання), як вони використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?
10. Що таке візуалізації, як вони використовуються в середовищі програмування CoDeSys фірми 3S (Smart Software Solutions)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №3.
# РОБОТА ІЗ ПРОЕКТОМ В СЕРЕДОВИЩІ ПРОГРАМУВАННЯ CODESYS

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи із проектом в середовищі програмування CoDeSys.

## 2. Порядок виконання лабораторної роботи

1. Ознайомитися з режимами роботи із проектом в середовищі програмування CoDeSys за допомогою меню.

В протокол занести (можна скорочено):

- параметри пункту меню **File** (виконує всі основні операції із файлами);
- параметри пункту меню **Edit** (виконує всі основні операції по редагуванню текстових і графічних даних);
- параметри пункту меню **Project** (виконує всі основні операції із проектом);
- параметри пункту меню **Insert** (вставка в проект певних елементів в залежності від того, що саме редагується);
- параметри пункту меню **Extras** (дозволяє використовувати в проекті різні додаткові можливості);
- параметри пункту меню **Online** (виконує всі основні операції по запуску, зупинці, налагодженні, емуляції тощо програмного коду);
- параметри пункту меню **Window** (виконує всі основні операції із вікнами);
- параметри пункту меню **Help** (виконує всі основні операції по отриманню довідкової інформації по середовищу програмування CoDeSys).

2. Ознайомитися з режимами роботи із проектом в середовищі програмування CoDeSys за допомогою панелі інструментів.

В протокол занести (можна скорочено):

- параметри кнопки **New** (створює новий проект);
- параметри кнопки **Open** (відкриває існуючий проект);
- параметри кнопки **Save** (зберігає існуючий проект);
- параметри кнопки **Run** (запускає програмний код на виконання);
- параметри кнопки **Stop** (зупиняє виконання програмного коду);
- параметри кнопки **Step over** (виконує за один раз програмний код тільки одного компоненту організації програм);
- параметри кнопки **Toggle breakpoint** (встановлює або скидає точку зупинки в програмному коді);
- параметри кнопки **Login** (встановлення з'єднання з програмованим логічним контролером);
- параметри кнопки **Logout** (розірвання з'єднання з програмованим логічним контролером);

- параметри кнопки **Global search…** (пошук тексту в усьому проекті) і так далі.

3. Ознайомитися з пунктом меню **File**.

В протокол занести (можна скорочено):

- параметри команди **New** (створює новий проект);
- параметри команди **New from template** (створює новий проект на основі шаблону проекту);
- параметри команди **Open** (відкриває існуючий проект);
- параметри команди **Save** (зберігає існуючий проект);
- параметри команди **Save as…** (зберігає існуючий проект з новим іменем);
- параметри команди **Save/Mail Archive…** (зберігає існуючий проект в вигляді архіву);
- параметри команди **Print** (дозволяє надрукувати на принтері уміст поточного вікна);
- параметри команди **Print Setup…** (дозволяє задати параметри друку на принтері умісту поточного вікна);
- параметри команди **Exit** (завершує роботи з середовищем програмування CoDeSys).

4. Ознайомитися з пунктом меню **Project**.

В протокол занести (можна скорочено):

- параметри команди **Build** (дозволяє здійснити компіляцію проекту);
- параметри команди **Rebuild all** (дозволяє здійснити компіляцію всіх компонентів організації програм проекту);
- параметри команди **Clean all** (видаляє всю інформацію про попередню компіляцію і про завантаження проекту в програмований логічний контролер);
- параметри команди **Load download information…** (завантажує інформацію про завантаження програмного коду в програмований логічний контролер);
- параметри команди **Options…** (дозволяє здійснити налагодження проекту);
- параметри команди **Document…** (дозволяє надрукувати на принтері уміст поточного проекту);
- параметри команди **Export…** (дозволяє здійснити експорт проекту);
- параметри команди **Import…** (дозволяє здійснити імпорт проекту).

5. Ознайомитися з елементами пункту меню **Project–Options…**.

В протокол занести (можна скорочено):

- параметри вкладки **Load & Save** вікна **Options** (дозволяє контролювати завантаження і збереження файлів);
- параметри вкладки **User information** вікна **Options** (дозволяє контролювати інформацію про програміста);
- параметри вкладки **Editor** вікна **Options** (дозволяє контролювати параметри редагування);
- параметри вкладки **Desktop** вікна **Options** (дозволяє контролювати параметри робочого столу);

- параметри вкладки **Colors** вікна **Options** (дозволяє контролювати колір елементів оформлення програмного коду);

- параметри вкладки **Directories** вікна **Options** (дозволяє контролювати розташування файлів);

- параметри вкладки **Log** вікна **Options** (дозволяє контролювати параметри бортжурналу);

- параметри вкладки **Build** вікна **Options** (дозволяє контролювати параметри компіляції програмного коду);

- параметри вкладки **Passwords** вікна **Options** (дозволяє контролювати паролі);

- параметри вкладки **Source download** вікна **Options** (дозволяє контролювати завантаження файлів в програмований логічний контролер);

- параметри вкладки **Symbol configuration** вікна **Options** (дозволяє контролювати обмін даними з програмованим логічним контролером);

- параметри вкладки **Database connection** вікна **Options** (дозволяє контролювати обмін даними з базою даних);

- параметри вкладки **Macros** вікна **Options** (дозволяє контролювати порядок використання макрокоманд).

### 3. Питання для самоперевірки

1. Як здійснюється робота з проектом в середовищі програмування CoDeSys?
2. Які основні елементи має меню головного вікна середовища програмування CoDeSys?
3. Які основні елементи має панель інструментів головного вікна середовища програмування CoDeSys?
4. Які основні елементи має організатор об’єктів головного вікна середовища програмування CoDeSys?
5. Які основні елементи має робоча область редактора головного вікна середовища програмування CoDeSys?
6. Які основні елементи має вікно повідомлень головного вікна середовища програмування CoDeSys?
7. Які основні елементи має рядок статусу головного вікна середовища програмування CoDeSys?
8. Які основні елементи, що дозволяють керувати проектом, має пункт меню **File** середовища програмування CoDeSys?
9. Які основні елементи, що дозволяють керувати проектом, має пункт меню **Project** середовища програмування CoDeSys?
10. Які основні елементи, що дозволяють налагоджувати проект, має пункту меню **Project–Options** середовища програмування CoDeSys?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №4.
# СТВОРЕННЯ ПРОГРАМНОГО КОДУ В СЕРЕДОВИЩІ ПРОГРАМУВАННЯ CODESYS

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення створення програмного коду в середовищі програмування CoDeSys.

## 2. Порядок виконання лабораторної роботи

1. Ознайомитися з редакторами програмних компонентів (розділ коду і розділ об’яв) в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- параметри розділу коду;
- параметри розділу об’яв.

2. Ознайомитися з правильним розбиттям програмного коду на листи при друку на принтері в середовищі програмування CoDeSys.

В протокол занести (можна скорочено): призначення прапорця **Show print area margins**, який розташований на вкладці **Desktop** вікна **Options** пункту меню **Project–Options…**.

3. Ознайомитися з порядком розміщення в програмному коді коментарів в середовищі програмування CoDeSys.

В протокол занести (можна скорочено): порядок розміщення коментарів в розділі об’яв і в розділі коду текстових і графічних редакторів.

4. Ознайомитися з інтелектуальним уведенням в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- параметри інтелектуального уведення в розділі об’яв;
- параметри інтелектуального уведення в розділі коду.

5. Ознайомитися з порядком опису вхідних змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису вхідних змінних.

6. Ознайомитися з порядком опису вихідних змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису вихідних змінних.

7. Ознайомитися з порядком опису вхідних/вихідних змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису вхідних/вихідних змінних.

8. Ознайомитися з порядком опису локальних змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису локальних змінних.

9. Ознайомитися з порядком опису RETAIN-змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису RETAIN-змінних.

10. Ознайомитися з порядком опису PERSISTENT-змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису PERSISTENT-змінних.

11. Ознайомитися з порядком опису констант в середовищі програмування CoDeSys.

В протокол занести: порядок опису констант.

12. Ознайомитися з порядком опису EXTERNAL-змінних в середовищі програмування CoDeSys.

В протокол занести: порядок опису EXTERNAL-змінних.

13. Ознайомитися з існуючими зарезервованими словами в середовищі програмування CoDeSys.

В протокол занести: кілька зарезервованих слів (до кожного зарезервованого слова вказати його основне призначення).

14. Ознайомитися з порядком об’яви змінних в програмному коді в середовищі програмування CoDeSys.

В протокол занести: порядок об’яви змінних в програмному коді.

15. Ознайомитися з директивами компілятора в середовищі програмування CoDeSys.

В протокол занести: кілька директив компілятора (до кожної директиви компілятора вказати її основне призначення).

16. Ознайомитися з арифметичними операторами в середовищі програмування CoDeSys.

В протокол занести: кілька арифметичних операторів (до кожного арифметичного оператора вказати його основне призначення).

17. Ознайомитися з бітовими операторами в середовищі програмування CoDeSys.

В протокол занести: кілька бітових операторів (до кожного бітового оператора вказати його основне призначення).

18. Ознайомитися з операторами зсуву в середовищі програмування CoDeSys.

В протокол занести: кілька операторів зсуву (до кожного оператора зсуву вказати його основне призначення).

19. Ознайомитися з операторами вибірки в середовищі програмування CoDeSys.

В протокол занести: кілька операторів вибірки (до кожного оператора вибірки вказати його основне призначення).

20. Ознайомитися з операторами порівняння в середовищі програмування CoDeSys.

В протокол занести: кілька операторів порівняння (до кожного оператора порівняння вказати його основне призначення).

21. Ознайомитися з адресними операторами в середовищі програмування CoDeSys.

В протокол занести: кілька адресних операторів (до кожного адресного оператора вказати його основне призначення).

22. Ознайомитися з операторами виклику в середовищі програмування

CoDeSys.

В протокол занести: кілька операторів виклику (до кожного оператора виклику вказати його основне призначення).

23. Ознайомитися з допоміжними функціями в середовищі програмування CoDeSys.

В протокол занести: кілька допоміжних функцій (до кожної допоміжної функції вказати її основне призначення).

23. Ознайомитися з функціями явного перетворення типів в середовищі програмування CoDeSys.

В протокол занести: кілька функцій явного перетворення типів (до кожної функції явного перетворення типів вказати її основне призначення).

24. Ознайомитися з математичними функціями в середовищі програмування CoDeSys.

В протокол занести: кілька математичних функцій (до кожної математичної функції вказати її основне призначення).

### 3. Питання для самоперевірки

1. Як здійснюється створення програмного коду в середовищі програмування CoDeSys?
2. Що таке редактори програмних компонентів середовища програмування CoDeSys?
3. Що таке межі друкованого листа середовища програмування CoDeSys?
4. Що таке коментарі середовища програмування CoDeSys?
5. Що таке інтелектуальне уведення середовища програмування CoDeSys?
6. Що таке розділ об’яв середовища програмування CoDeSys?
7. Що таке зарезервовані слова середовища програмування CoDeSys?
8. Що таке об’ява змінних в програмному коді середовища програмування CoDeSys?
9. Що таке директиви компілятора в програмному коді середовища програмування CoDeSys?
10. Які арифметичні оператори є в середовищі програмування CoDeSys?
11. Які бітові оператори є в середовищі програмування CoDeSys?
12. Які оператори зсуву є в середовищі програмування CoDeSys?
13. Які оператори вибірки є в середовищі програмування CoDeSys?
14. Які оператори порівняння є в середовищі програмування CoDeSys?
15. Які адресні оператори є в середовищі програмування CoDeSys?
16. Які оператори виклику є в середовищі програмування CoDeSys?
17. Які допоміжні функції, функції явного перетворення типів і математичні функції є в середовищі програмування CoDeSys?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №5.
# ОСНОВНІ РЕСУРСИ СЕРЕДОВИЩА ПРОГРАМУВАННЯ CODESYS

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення основних ресурсів середовища програмування CoDeSys.

## 2. Порядок виконання лабораторної роботи

1. Ознайомитися з *Глобальними змінними* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок визначення *Глобальних змінних*;
- порядок структурування *Глобальних змінних*;
- порядок створення і редагування списків *Глобальних змінних*.

2. Ознайомитися з *Конфігурацією тривог* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- визначення сигнальної системи;
- визначення класів тривог;
- визначення груп тривог;
- порядок запису тривог.

3. Ознайомитися з *Менеджером бібліотек* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок отримання інформації про бібліотеки;
- уміст бібліотеки standard.lib;
- порядок створення бібліотеки.

4. Ознайомитися з *Бортжурналом* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок створення протоколу послідовності дій під час Online-сесії;
- порядок визначення категорії і опису дій;
- порядок збереження протоколу послідовності дій в режимі Online.

5. Ознайомитися з *Конфігурацією ПЛК* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок конфігурації модулів уведення/виведення;
- порядок конфігурації каналів;
- порядок роботи *Конфігурації ПЛК* в режимі Online;
- порядок визначення стану модулів уведення/виведення.

6. Ознайомитися з *Менеджером перегляду* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок роботи в редакторі *Менеджера перегляду* в режимі Offline;

- порядок роботи в редакторі *Менеджера перегляду* в режимі Online.

7. Ознайомитися з *Конфігурацією задач* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок роботи в редакторі *Конфігурації задач*;
- параметри системних повідомлень.

8. Ознайомитися з *Налагодженнями цільової платформи* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено): параметри, які визначають архітектуру мікропроцесора, розподіл областей пам'яті програмованого логічного контролера, загальні налагодження програмованого логічного контролера, мережеві налагодження програмованого логічного контролера, налагодження візуалізацій програмованого логічного контролера.

9. Ознайомитися з *Робочою областю* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено): параметри налагодження *Робочої області*.

### 3. Питання для самоперевірки

1. Які основні ресурси має середовище програмування CoDeSys, як ці ресурси використовуються?
2. Що таке і як застосовуються *Глобальні змінні* в середовищі програмування CoDeSys?
3. Що таке і як працює *Конфігурація тривог* в середовищі програмування CoDeSys?
4. Що таке і як працює *Менеджер бібліотек* в середовищі програмування CoDeSys?
5. Що таке і як працює *Бортжурнал* в середовищі програмування CoDeSys?
6. Що таке і як працює *Конфігурація ПЛК* в середовищі програмування CoDeSys?
7. Що таке і як працює *Менеджер перегляду* в середовищі програмування CoDeSys?
8. Що таке і як працює *Конфігурація задач* в середовищі програмування CoDeSys?
9. Що таке і як працює *Налагодження цільової платформи* в середовищі програмування CoDeSys?
10. Що таке і як працює *Робоча область* в середовищі програмування CoDeSys?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №6.
# ДОДАТКОВІ РЕСУРСИ СЕРЕДОВИЩА ПРОГРАМУВАННЯ CODESYS

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення додаткових ресурсів середовища програмування CoDeSys.

## 2. Порядок виконання лабораторної роботи

1. Ознайомитися з *Менеджером параметрів* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок підключення *Менеджера параметрів*;
- порядок роботи в редакторі *Менеджера параметрів*;
- визначення типів списків параметрів і їх атрибутів;
- порядок керування списками параметрів;
- порядок редагування списку параметрів;
- порядок роботи *Менеджера параметрів* в режимі Online;
- порядок експорту/імпорту списку параметрів.

2. Ознайомитися з *ПЛК-браузером* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- визначення набору команд;
- визначення макророзширень команд;
- визначення додаткових команд.

3. Ознайомитися з *Цифровим трасуванням* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- порядок конфігурації *Цифрового трасування*;
- порядок керування процесом цифрового трасування;
- порядок відображення даних.

4. Ознайомитися з *Інструментами* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- визначення властивостей доступних *Інструментів*;
- порядок конфігурації команд доступних *Інструментів*.

5. Ознайомитися з *Системою керування рухом SoftMotion* в середовищі програмування CoDeSys.

В протокол занести (можна скорочено):

- призначення компонента Drive interface;
- призначення компонента Configuration editor;
- призначення компонента Drive Interface libraries;
- призначення компонента CNC-editor;
- призначення компонента CAM-editor;
- призначення компонента CNC-libraries;
- призначення компонента PLCopen-library;

- призначення компонента File service library;
- призначення компонента Error library.

**3. Питання для самоперевірки**

1. Які додаткові ресурси має середовище програмування CoDeSys?
2. Які можливості має *Менеджер параметрів*, як здійснюється його підключення в середовищі програмування CoDeSys?
3. Які типи списків параметрів і їх атрибути має *Менеджер параметрів*, як здійснюється керування списками параметрів, редагування списку параметрів і експорт/імпорт списку параметрів в *Менеджері параметрів* в середовищі програмування CoDeSys?
4. Що таке *ПЛК-браузер* в середовищі програмування CoDeSys?
5. Які набір команд, макророзширення команд і додаткові команди має *ПЛК-браузер* в середовищі програмування CoDeSys?
6. Що таке *Цифрове трасування* в середовищі програмування CoDeSys?
7. Як здійснюється конфігурація *Цифрового трасування*, керування процесом цифрового трасування і відображення даних в середовищі програмування CoDeSys?
8. Що таке *Інструменти* в середовищі програмування CoDeSys?
9. Які властивості мають доступні *Інструменти*, як здійснюється конфігурація команд доступних *Інструментів* в середовищі програмування CoDeSys?
10. Що таке *Система керування рухом SoftMotion*?
11. Яке призначення має компонент Drive interface *Системи керування рухом SoftMotion*?
12. Яке призначення має компонент Configuration editor *Системи керування рухом SoftMotion*?
13. Яке призначення має компонент Drive Interface libraries *Системи керування рухом SoftMotion*?
14. Яке призначення має компонент CNC-editor *Системи керування рухом SoftMotion*?
15. Яке призначення має компонент CAM-editor *Системи керування рухом SoftMotion*?
16. Яке призначення має компонент CNC-libraries *Системи керування рухом SoftMotion*?
17. Яке призначення має компонент PLCopen-library *Системи керування рухом SoftMotion*?
18. Яке призначення має компонент File service library *Системи керування рухом SoftMotion*?
19. Яке призначення має компонент Error library *Системи керування рухом SoftMotion*?

**4. Рекомендована література**

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №7.
# МОВА ПРОГРАМУВАННЯ INSTRUCTION LIST (IL)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення мови програмування **Instruction List** (**IL**).

## 2. Порядок виконання лабораторної роботи

### 2.1. Вивчення операторів

#### 2.1.1. Вивчення операторів завантаження LD і ST

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **BOOL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      FALSE
ST      Z
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LDN     FALSE
ST      Z
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      FALSE
STN     Z
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LDN     FALSE
STN     Z
```

10. В режимі виконання програмного коду визначити значення змінної **Z**.

#### 2.1.2. Вивчення операторів завантаження S і R

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **BOOL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      FALSE
S       Z
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      TRUE
S       Z
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      FALSE
```

**R Z**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD TRUE**
**R Z**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.3. Вивчення арифметичних операторів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **DINT**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD 2**
**ADD 3**
**ST Z**

4. В режимі виконання програмного коду визначити значення змінної **Z**.

5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD 2**
**SUB 3**
**ST Z**

6. В режимі виконання програмного коду визначити значення змінної **Z**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD 60**
**MUL 3**
**ST Z**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD 60**
**DIV 3**
**ST Z**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD 10**
**MOD 2**
**ST Z**

12. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.4. Вивчення бітових операторів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **BYTE**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD 2#1001_0011**
**AND 2#1000_1010**
**ST Z**

4. В режимі виконання програмного коду визначити значення змінної **Z**.

5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      2#1001_0011
OR      2#1000_1010
ST      Z
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      2#1001_0011
XOR     2#1000_1010
ST      Z
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      2#1001_0011
NOT
ST      Z
```

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.5. Вивчення операторів зсуву**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **BYTE**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      16#54
SHL     2
ST      Z
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      16#54
SHR     2
ST      Z
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      16#54
ROL     2
ST      Z
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      16#54
ROR     2
ST      Z
```

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.6. Вивчення операторів вибірки**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **DINT**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      TRUE
SEL     10, 20
```

**ST** **Z**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **10**
**MAX** **20**
**ST** **Z**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **10**
**MIN** **20**
**ST** **Z**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **–50**
**LIMIT** **50, 100**
**ST** **Z**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **1**
**MUX** **10, 20, 30, 40, 50**
**ST** **Z**

12. В режимі виконання програмного коду визначити значення змінної **Z**.

### 2.1.7. Вивчення операторів порівняння

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **BOOL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **50**
**GT** **25**
**ST** **Z**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **35**
**LT** **70**
**ST** **Z**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **80**
**GE** **40**
**ST** **Z**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**LD** **45**
**LE** **90**
**ST** **Z**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      15
EQ      15
ST      Z
```

12. В режимі виконання програмного коду визначити значення змінної **Z**.
13. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      10
NE      20
ST      Z
```

14. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.8. Вивчення математичних операторів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **REAL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      –15
ABS
ST      Z
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      25
SQRT
ST      Z
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      50
LN
ST      Z
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      60
LOG
ST      Z
```

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      1.6
EXP
ST      Z
```

12. В режимі виконання програмного коду визначити значення змінної **Z**.
13. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      –5.5
SIN
ST      Z
```

14. В режимі виконання програмного коду визначити значення змінної **Z**.

15. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      –5.5
COS
ST      Z
```

16. В режимі виконання програмного коду визначити значення змінної **Z**.

17. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      1.4
TAN
ST      Z
```

18. В режимі виконання програмного коду визначити значення змінної **Z**.

19. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      –0.8
ASIN
ST      Z
```

20. В режимі виконання програмного коду визначити значення змінної **Z**.

21. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      –0.8
ACOS
ST      Z
```

22. В режимі виконання програмного коду визначити значення змінної **Z**.

23. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      4.6
ATAN
ST      Z
```

24. В режимі виконання програмного коду визначити значення змінної **Z**.

25. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      3
EXPT    5
ST      Z
```

26. В режимі виконання програмного коду визначити значення змінної **Z**.

### 2.2. Вивчення виклику функцій

#### 2.2.1. Вивчення виклику функцій з одним параметром

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **REAL**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      10
_F_
ST      Z
```

4. Створити функцію **_F_** в середовищі програмування CoDeSys на мові програмування **IL** із вхідною змінною **X** типу **REAL** і значенням, яке повертається, типу **REAL**.

5. Увести програмний код, наведений нижче, в функцію **_F_**.

```
LD      X
```

```
MUL     5
ST      _F_
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.2.2. Вивчення виклику функцій з двома параметрами**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **REAL**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      10
_F_     20
ST      Z
```

4. Створити функцію **_F_** в середовищі програмування CoDeSys на мові програмування **IL** із вхідною змінною **X1** типу **REAL**, вхідною змінною **X2** типу **REAL** і значенням, яке повертається, типу **REAL**.

5. Увести програмний код, наведений нижче, в функцію **_F_**.

```
LD      X1
ADD     X2
MUL     5
ST      _F_
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.3. Вивчення виклику функціональних блоків**

**2.3.1. Вивчення виклику функціональних блоків з попереднім присвоєнням значень вхідним змінним**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **REAL**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      10
ST      _FB_.X1
LD      20
ST      _FB_.X2
LD      30
ST      _FB_.X3
CAL     _FB_
LD      _FB_.Y
ST      Z
```

4. Створити функціональний блок **FB** в середовищі програмування CoDeSys на мові програмування **IL** із вхідною змінною **X1** типу **REAL**, вхідною змінною **X2** типу **REAL**, вхідною змінною **X3** типу **REAL** і вихідною змінною **Y** типу **REAL**.

5. Створити змінну **_FB_** типу **FB**.

6. Увести програмний код, наведений нижче, в функціональний блок **FB**.

```
LD      X1
ADD     X2
```

```
ADD     X3
MUL     5
ST      Y
```

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.3.2. Вивчення виклику функціональних блоків з присвоєнням значень вхідним змінним в момент виклику**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **REAL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
CAL     _FB_(X1:=10, X2:=20, X3:=30)
LD      _FB_.Y
ST      Z
```

4. Створити функціональний блок **FB** в середовищі програмування CoDeSys на мові програмування **IL** із вхідною змінною **X1** типу **REAL**, вхідною змінною **X2** типу **REAL**, вхідною змінною **X3** типу **REAL** і вихідною змінною **Y** типу **REAL**.
5. Створити змінну **_FB_** типу **FB**.
6. Увести програмний код, наведений нижче, в функціональний блок **FB**.

```
LD      X1
ADD     X2
ADD     X3
MUL     5
ST      Y
```

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.4. Вивчення виклику програм**

**2.4.1. Вивчення виклику програм з попереднім присвоєнням значень вхідним змінним**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).
2. Створити змінну **Z** типу **REAL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      10
ST      _PRG_.X1
LD      20
ST      _PRG_.X2
LD      30
ST      _PRG_.X3
CAL     _PRG_
LD      _PRG_.Y
ST      Z
```

4. Створити програму **_PRG_** в середовищі програмування CoDeSys на мові програмування **IL** із вхідною змінною **X1** типу **REAL**, вхідною змінною **X2** типу **REAL**, вхідною змінною **X3** типу **REAL** і вихідною змінною **Y** типу

**REAL**.

5. Увести програмний код, наведений нижче, в програму **_PRG_**.

```
LD      X1
ADD     X2
ADD     X3
MUL     5
ST      Y
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.4.2. Вивчення виклику програм з присвоєнням значень вхідним змінним в момент виклику**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **REAL**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
CAL     _PRG_(X1:=10, X2:=20, X3:=30)
LD      _PRG_.Y
ST      Z
```

4. Створити програму **_PRG_** в середовищі програмування CoDeSys на мові програмування **IL** із вхідною змінною **X1** типу **REAL**, вхідною змінною **X2** типу **REAL**, вхідною змінною **X3** типу **REAL** і вихідною змінною **Y** типу **REAL**.

5. Увести програмний код, наведений нижче, в програму **_PRG_**.

```
LD      X1
ADD     X2
ADD     X3
MUL     5
ST      Y
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5. Вивчення застосування дужок в виразах**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **IL**).

2. Створити змінну **Z** типу **REAL**.

3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      2
ADD     (3
MUL     4
)
ST      Z
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.

5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      2
ADD     (3
MUL     (4
```

```
SUB     1
)
)
ST      Z
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      2
ADD     (3
MUL     (4
SUB     (1
DIV     0.2
)
)
)
ST      Z
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.

### 3. Питання для самоперевірки

1. Що таке і які особливості має мова програмування **Instruction List** (**IL**)?
2. Який формат інструкцій застосовується в мові програмування **Instruction List** (**IL**)?
3. Як в мові програмування **Instruction List** (**IL**) застосовується акумулятор?
4. Як в мові програмування **Instruction List** (**IL**) здійснюється перехід на мітку?
5. Як в мові програмування **Instruction List** (**IL**) застосовуються скобки?
6. Як в мові програмування **Instruction List** (**IL**) застосовуються модифікатори?
7. Які оператори застосовуються в мові програмування **Instruction List** (**IL**)?
8. Як здійснюється виклик функцій в мові програмування **Instruction List** (**IL**)?
9. Як здійснюється виклик функціональних блоків в мові програмування **Instruction List** (**IL**)?
10. Як здійснюється виклик програм в мові програмування **Instruction List** (**IL**)?
11. Як здійснюється коментування тексту в мові програмування **Instruction List** (**IL**)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №8.
# МОВА ПРОГРАМУВАННЯ STRUCTURED TEXT (ST)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення мови програмування **Structured Text** (**ST**).

## 2. Порядок виконання лабораторної роботи

### 2.1. Вивчення виразів

#### 2.1.1. Вивчення виразів з арифметичними операторами

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **DINT**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 10+20 ;
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 30–5 ;
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 3*6 ;
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 60/4 ;
```

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 82 MOD 10 ;
```

12. В режимі виконання програмного коду визначити значення змінної **Z**.

#### 2.1.2. Вивчення виразів з бітовими операторами

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **BYTE**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 2#1001_0001 AND 2#1000_1011 ;
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 2#1001_0001 OR 2#1000_1011 ;
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 2#1001_0001 XOR 2#1000_1011 ;
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := NOT 2#1001_0011 ;
```

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.3. Вивчення виразів з операторами зсуву**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **BYTE**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := SHL(16#56,2) ;**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := SHR(16#56,2) ;**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := ROL(16#56,2) ;**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := ROR(16#56,2) ;**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.4. Вивчення виразів з операторами вибірки**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **DINT**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := SEL(TRUE,15,30) ;**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := MAX(15,30) ;**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := MIN(15,30) ;**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := LIMIT(–10,45,30) ;**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := MUX(2,10,20,30,40,50) ;**

12. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.5. Вивчення виразів з операторами порівняння**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **BOOL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := 60>30 ;**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := 40<70 ;**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := 90>=50 ;**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := 50<=100 ;**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := 20=20 ;**

12. В режимі виконання програмного коду визначити значення змінної **Z**.
13. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := 45<>45 ;**

14. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.6. Вивчення виразів з математичними операторами**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **REAL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := ABS(–18) ;**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := SQRT(36) ;**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := LN(250) ;**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := LOG(380) ;**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := EXP(2.3) ;**

12. В режимі виконання програмного коду визначити значення змінної **Z**.
13. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := SIN(–6.5) ;**

14. В режимі виконання програмного коду визначити значення змінної **Z**.
15. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := COS(–4.9) ;**

16. В режимі виконання програмного коду визначити значення змінної **Z**.
17. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := TAN(3.3) ;**

18. В режимі виконання програмного коду визначити значення змінної **Z**.
19. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := ASIN(–0.94) ;**

20. В режимі виконання програмного коду визначити значення змінної **Z**.
21. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := ACOS(0.77) ;**

22. В режимі виконання програмного коду визначити значення змінної **Z**.
23. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := ATAN(12.5) ;**

24. В режимі виконання програмного коду визначити значення змінної **Z**.
25. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**Z := EXPT(3,6) ;**

26. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.2. Вивчення оператора вибору IF**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **REAL**.
3. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
IF 10>5
THEN
        Z := 100 ;
END_IF
```

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
IF 14>20
THEN
        Z := 150 ;
ELSE
        Z := 450 ;
END_IF
```

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
IF 50<30
THEN
        Z := 12 ;
ELSIF 20>10
THEN
        Z := 18 ;
ELSE
        Z := 22 ;
END_IF
```

8. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.3. Вивчення оператора множинного вибору CASE**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **REAL**.
3. Створити змінну **I** типу **BYTE**.
4. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

**I := 5 ;**

```
CASE I OF
5 :
        Z := 10 ;
10,20,30,40,50 :
        Z := 20 ;
END_CASE
```

5. В режимі виконання програмного коду визначити значення змінної **Z**.

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
I := 5 ;
CASE I OF
10 :
        Z := 15 ;
20..30 :
        Z := 30 ;
ELSE
        Z := 45 ;
END_CASE
```

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.4. Вивчення оператора циклу WHILE**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).

2. Створити змінну **Z** типу **REAL**.

3. Створити змінну **I** типу **BYTE**.

4. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 0 ;
I := 0 ;
WHILE I<10 DO
        Z := Z+I*I ;
        I := I+1 ;
END_WHILE
```

5. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5. Вивчення оператора циклу REPEAT**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).

2. Створити змінну **Z** типу **REAL**.

3. Створити змінну **I** типу **BYTE**.

4. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 0 ;
I := 30 ;
REPEAT
        Z := Z+I ;
        I := I–1 ;
UNTIL I<5
END_REPEAT
```

5. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.6. Вивчення оператора циклу FOR**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **ST**).
2. Створити змінну **Z** типу **REAL**.
3. Створити змінну **I** типу **BYTE**.
4. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 0 ;
FOR I:=1 TO 10 DO
        Z := Z+LN(I) ;
END_FOR
```

5. В режимі виконання програмного коду визначити значення змінної **Z**.
6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
Z := 0 ;
FOR I:=10 TO 100 BY 5 DO
        Z := Z+LOG(I) ;
END_FOR
```

**3. Питання для самоперевірки**

1. Що таке і які особливості має мова програмування **Structured Text** (**ST**)?
2. Як здійснюється порядок обчислення виразів в мові програмування **Structured Text** (**ST**)?
3. Що таке пустий оператор в мові програмування **Structured Text** (**ST**)?
4. Що таке оператор присвоєння в мові програмування **Structured Text** (**ST**)?
5. Як застосовується в мові програмування **Structured Text** (**ST**) оператор вибору **IF**?
6. Як застосовується в мові програмування **Structured Text** (**ST**) оператор множинного вибору **CASE**?
7. Як застосовується в мові програмування **Structured Text** (**ST**) оператор циклу **WHILE**?
8. Як застосовується в мові програмування **Structured Text** (**ST**) оператор циклу **REPEAT**?
9. Як застосовується в мові програмування **Structured Text** (**ST**) оператор циклу **FOR**?
10. Як застосовується в мові програмування **Structured Text** (**ST**) переривання ітерацій оператором **EXIT**?
11. Як застосовується в мові програмування **Structured Text** (**ST**) переривання виконання програмного коду оператором **RETURN**?

**4. Рекомендована література**

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №9.
# МОВА ПРОГРАМУВАННЯ SEQUENTIAL FUNCTION CHART (SFC)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення мови програмування **Sequential Function Chart** (**SFC**).

## 2. Порядок виконання лабораторної роботи

### 2.1. Вивчення кроків і переходів

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).
2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.
3. Створити змінну **I** типу **INT** з початковим значенням **0**.
4. Створити **SFC**-схему (див. рис. 9.1). При її створенні не використовувати МЕК-кроки.

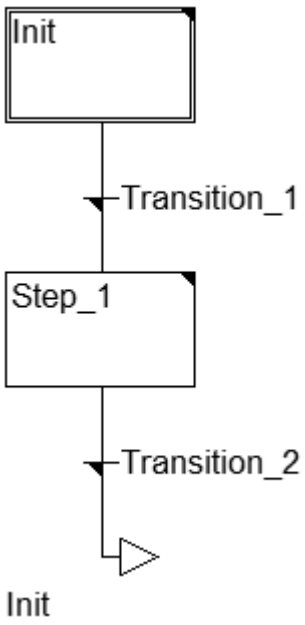



Рис. 9.1. Кроки і переходи

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

```
I := I+1 ;
```

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

```
I >= 200
```

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Step_1**.

```
I := 0 ;
Z := Z+1 ;
```

8. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**TRUE**

9. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.2. Вивчення паралельних гілок**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z1** типу **REAL** з початковим значенням **0**.

3. Створити змінну **Z2** типу **REAL** з початковим значенням **0**.

4. Створити **SFC**-схему (див. рис. 9.2). При її створенні не використовувати МЕК-кроки.

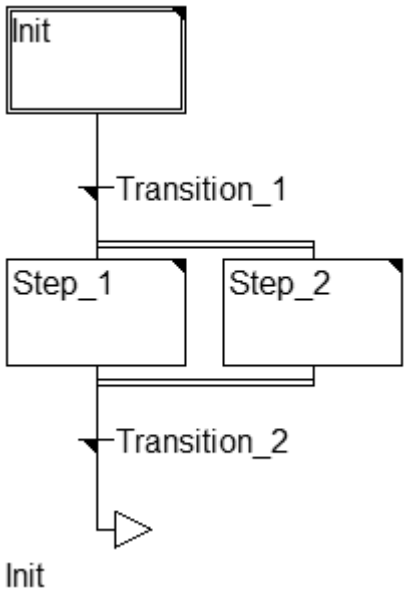



Рис. 9.2. Паралельні гілки

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Step_1**.

**Z1 := Z1+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Step_2**.

**Z2 := Z2-1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**TRUE**

10. В режимі виконання програмного коду визначити значення змінних **Z1** і **Z2**.

**2.3. Вивчення альтернативних гілок**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.3). При її створенні не використовувати МЕК-кроки.

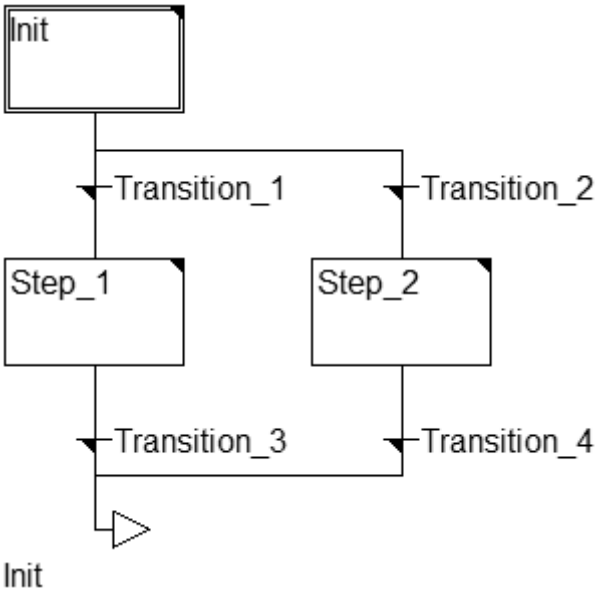



Рис. 9.3. Альтернативні гілки

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 5 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**I<=10**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I>10**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Step_1**.

**Z1 := Z1+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Step_2**.

**Z2 := Z2-1 ;**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**TRUE**

11. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_4**.

**TRUE**

12. В режимі виконання програмного коду визначити значення змінної **Z**.

13. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 15 ;**

14. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.4. Вивчення переходу на довільний крок**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити **SFC**-схему (див. рис. 9.4). При її створенні не використовувати МЕК-кроки.

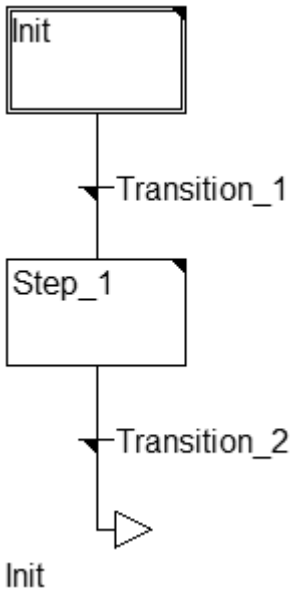



Рис. 9.4. Перехід на крок **Init**

4. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**Z := 0 ;**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Step_1**.

**Z1 := Z1+1 ;**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**TRUE**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Замінити на **SFC**-схемі перехід на крок **Init** переходом на крок **Step_1** (див. рис. 9.5).

10. В режимі виконання програмного коду визначити значення змінної **Z**.

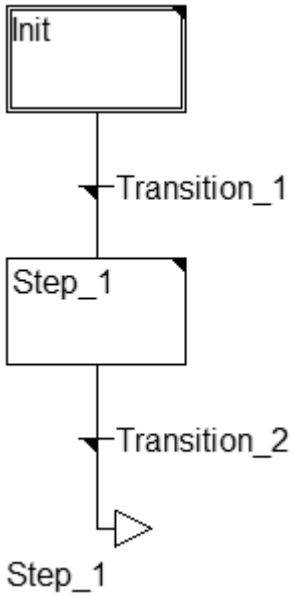



Рис. 9.5. Перехід на крок **Step_1**

**2.5. Вивчення класифікаторів дій**

**2.5.1. Вивчення класифікатора дій N**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.6). При її створенні використовувати МЕК-кроки.

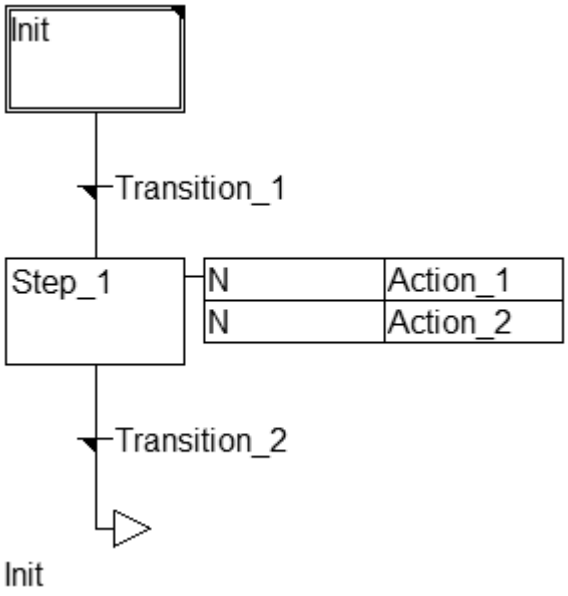



Рис. 9.6. Класифікатор дій **N**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

```
I := 0 ;
```

6. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 200**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.2. Вивчення класифікатора дій P**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.7). При її створенні використовувати МЕК-кроки.

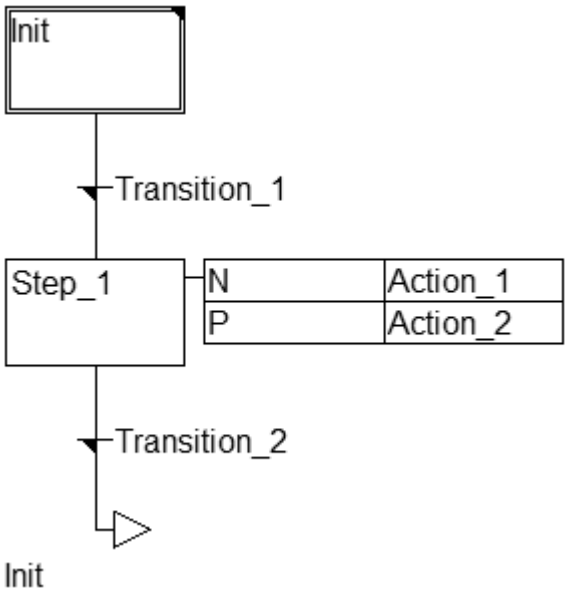



Рис. 9.7. Класифікатор дій **P**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 200**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.3. Вивчення класифікатора дій S**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.8). При її створенні використовувати МЕК-кроки.

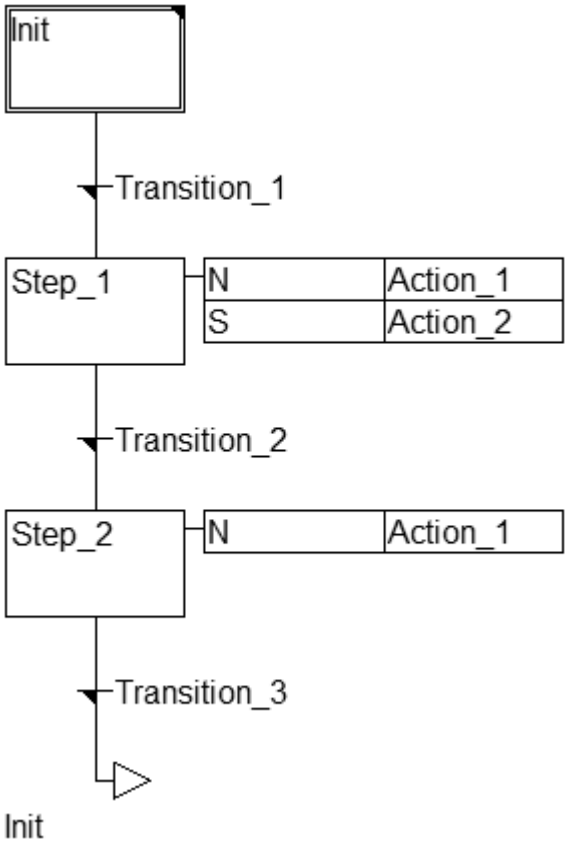



Рис. 9.8. Класифікатор дій **S**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 200**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**I >= 400**

11. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.4. Вивчення класифікатора дій R**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.9). При її створенні використовувати МЕК-кроки.

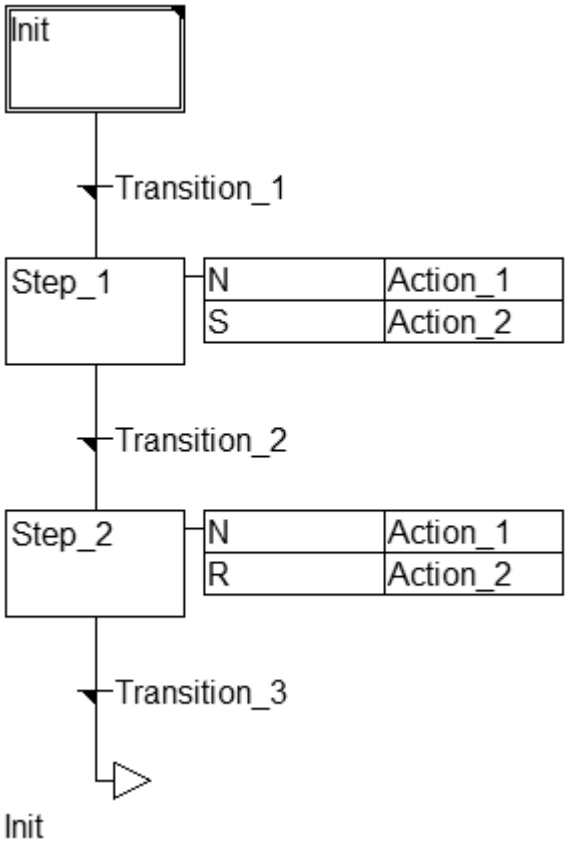



Рис. 9.9. Класифікатор дій **R**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 200**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**I >= 400**

11. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.5. Вивчення класифікатора дій L**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.10). При її створенні використовувати МЕК-кроки.

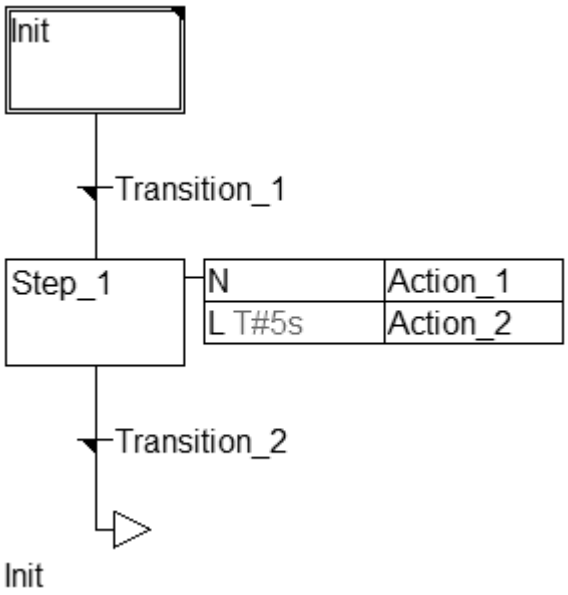



Рис. 9.10. Класифікатор дій **L**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 200**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.6. Вивчення класифікатора дій SL**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.11). При її створенні використовувати МЕК-кроки.

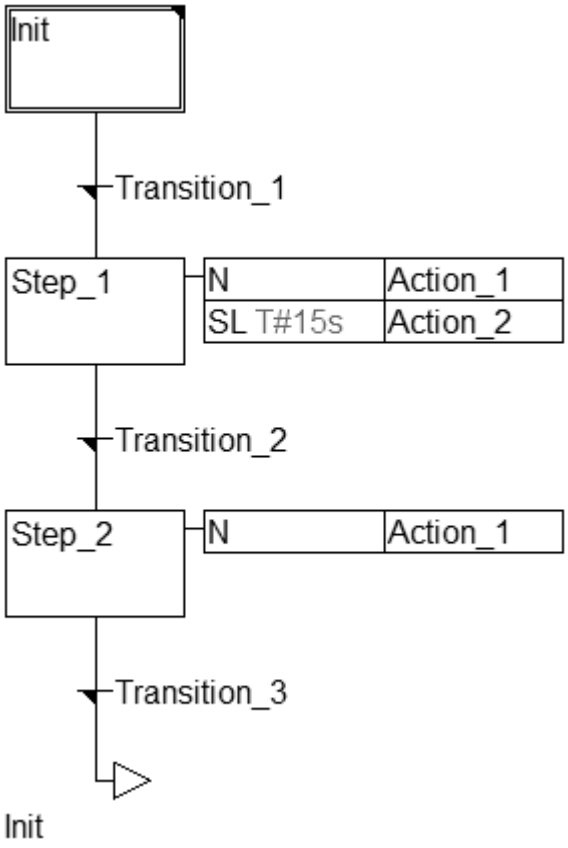



Рис. 9.11. Класифікатор дій **SL**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 500**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**I >= 1000**

11. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.7. Вивчення класифікатора дій D**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.12). При її створенні використовувати МЕК-кроки.

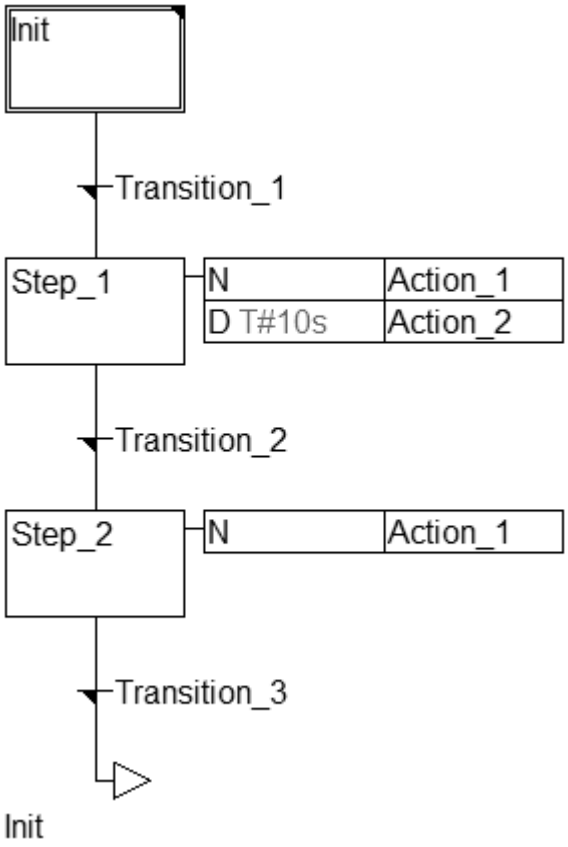



Рис. 9.12. Класифікатор дій **D**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 500**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**I >= 1000**

11. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.8. Вивчення класифікатора дій DS**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.13). При її створенні використовувати МЕК-кроки.

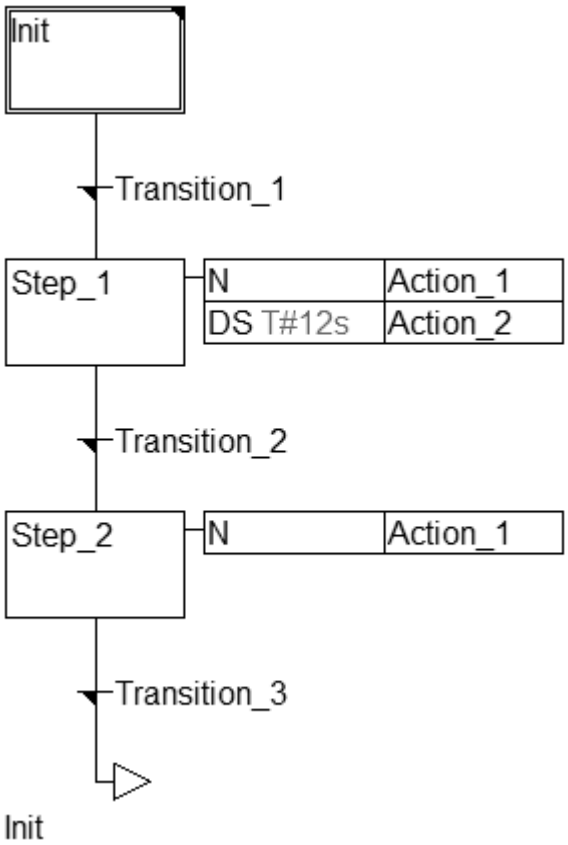



Рис. 9.13. Класифікатор дій **DS**

5. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 500**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**I >= 1000**

11. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5.9. Вивчення класифікатора дій SD**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **SFC**).

2. Створити змінну **Z** типу **REAL** з початковим значенням **0**.

3. Створити змінну **I** типу **INT**.

4. Створити **SFC**-схему (див. рис. 9.14). При її створенні використовувати МЕК-кроки.

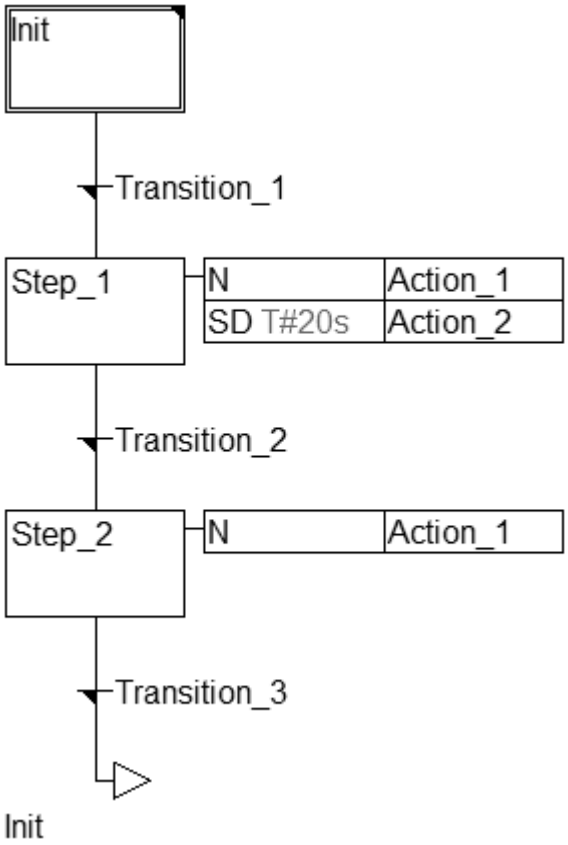



Рис. 9.14. Класифікатор дій **SD**

5. Увести програмний код, написаний на мові програмування **Structured**

**Text** (**ST**) і наведений нижче, в крок **Init**.

**I := 0 ;**

6. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_1**.

**TRUE**

7. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_1**.

**I := I+1 ;**

8. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в дію **Action_2**.

**Z := Z+1 ;**

9. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_2**.

**I >= 100**

10. Увести програмний код, написаний на мові програмування **Structured Text** (**ST**) і наведений нижче, в перехід **Transition_3**.

**I >= 1000**

11. В режимі виконання програмного коду визначити значення змінної **Z**.

### 3. Питання для самоперевірки

1. Що таке і які особливості має мова програмування **Sequential Function Chart** (**SFC**)?
2. Як застосовуються в мові програмування **Sequential Function Chart** (**SFC**) кроки і переходи?
3. Що таке початковий крок в мові програмування **Sequential Function Chart** (**SFC**)?
4. Що таке паралельні гілки і альтернативні гілки в мові програмування **Sequential Function Chart** (**SFC**)?
5. Як здійснюється перехід на довільний крок в мові програмування **Sequential Function Chart** (**SFC**)?
6. Що таке спрощена **SFC**-схема і що таке стандартна **SFC**-схема в мові програмування **Sequential Function Chart** (**SFC**)?
7. Що таке класифікатори дій в мові програмування **Sequential Function Chart** (**SFC**)?
8. Що таке дія-змінна в мові програмування **Sequential Function Chart** (**SFC**)?
9. Який механізм керування дією існує в мові програмування **Sequential Function Chart** (**SFC**)?
10. Що таке дія-змінна в мові програмування **Sequential Function Chart** (**SFC**)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №10.
# МОВА ПРОГРАМУВАННЯ FUNCTION BLOCK DIAGRAM (FBD)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення мови програмування **Function Block Diagram** (**FBD**).

## 2. Порядок виконання лабораторної роботи

### 2.1. Вивчення блоків

#### 2.1.1. Вивчення блоків з арифметичними операторами

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **DINT**.
3. Створити **FBD**-схему (див. рис. 10.1).

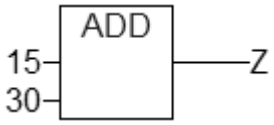



Рис. 10.1. Арифметичний оператор **ADD**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **FBD**-схему (див. рис. 10.2).

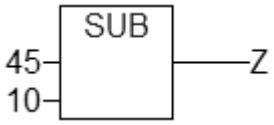



Рис. 10.2. Арифметичний оператор **SUB**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Змінити **FBD**-схему (див. рис. 10.3).

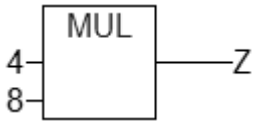



Рис. 10.3. Арифметичний оператор **MUL**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Змінити **FBD**-схему (див. рис. 10.4).

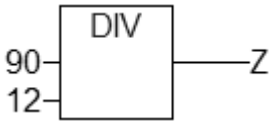



Рис. 10.4. Арифметичний оператор **DIV**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Змінити **FBD**-схему (див. рис. 10.5).

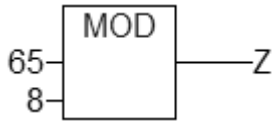



Рис. 10.5. Арифметичний оператор **MOD**

12. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.2. Вивчення блоків з бітовими операторами**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **BYTE**.
3. Створити **FBD**-схему (див. рис. 10.6).

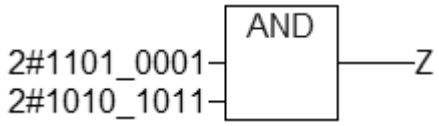



Рис. 10.6. Бітовий оператор **AND**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Створити **FBD**-схему (див. рис. 10.7).

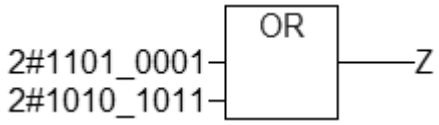



Рис. 10.7. Бітовий оператор **OR**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Створити **FBD**-схему (див. рис. 10.8).

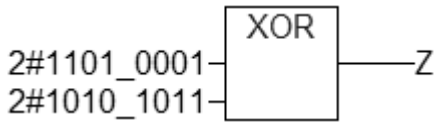



Рис. 10.8. Бітовий оператор **XOR**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Створити **FBD**-схему (див. рис. 10.9).

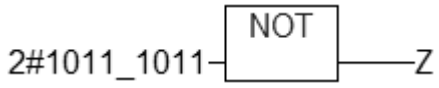



Рис. 10.9. Бітовий оператор **NOT**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.3. Вивчення блоків з операторами зсуву**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **BYTE**.
3. Створити **FBD**-схему (див. рис. 10.10).
4. В режимі виконання програмного коду визначити значення змінної **Z**.

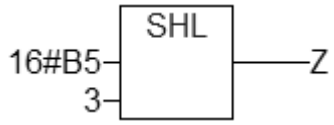



Рис. 10.10. Бітовий оператор **SHL**

5. Створити **FBD**-схему (див. рис. 10.11).

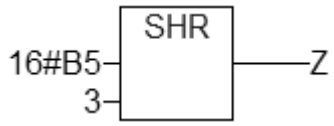



Рис. 10.11. Бітовий оператор **SHR**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Створити **FBD**-схему (див. рис. 10.12).

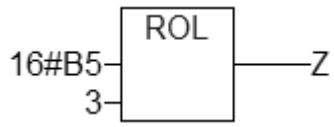



Рис. 10.12. Бітовий оператор **ROL**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Створити **FBD**-схему (див. рис. 10.13).

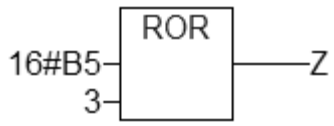



Рис. 10.13. Бітовий оператор **ROR**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.4. Вивчення блоків з операторами вибірки**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **DINT**.
3. Створити **FBD**-схему (див. рис. 10.14).

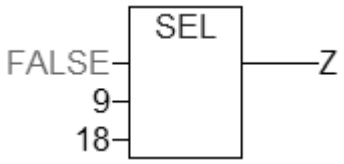



Рис. 10.14. Оператор вибірки **SEL**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Створити **FBD**-схему (див. рис. 10.15).

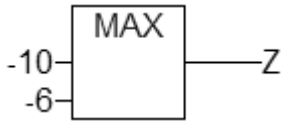



Рис. 10.15. Оператор вибірки **MAX**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Створити **FBD**-схему (див. рис. 10.16).

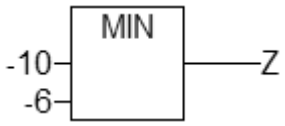



Рис. 10.16. Оператор вибірки **MIN**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Створити **FBD**-схему (див. рис. 10.17).

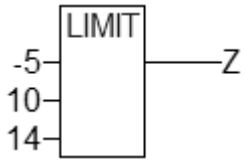



Рис. 10.17. Оператор вибірки **LIMIT**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Створити **FBD**-схему (див. рис. 10.18).

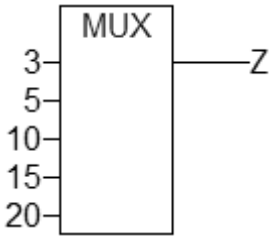



Рис. 10.18. Оператор вибірки **MUX**

12. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.5. Вивчення блоків з операторами порівняння**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити **FBD**-схему (див. рис. 10.19).

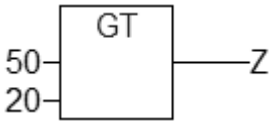



Рис. 10.19. Оператор порівняння **GT**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Створити **FBD**-схему (див. рис. 10.20).

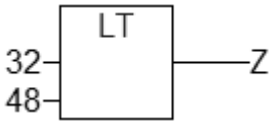



Рис. 10.20. Оператор порівняння **LT**

6. В режимі виконання програмного коду визначити значення змінної **Z**.

7. Створити **FBD**-схему (див. рис. 10.21).

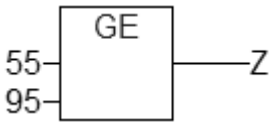



Рис. 10.21. Оператор порівняння **GE**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Створити **FBD**-схему (див. рис. 10.22).

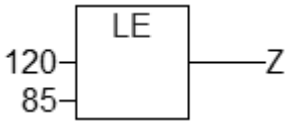



Рис. 10.22. Оператор порівняння **LE**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Створити **FBD**-схему (див. рис. 10.23).

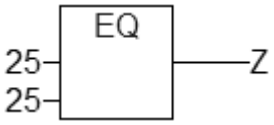



Рис. 10.23. Оператор порівняння **EQ**

12. В режимі виконання програмного коду визначити значення змінної **Z**.
13. Створити **FBD**-схему (див. рис. 10.24).

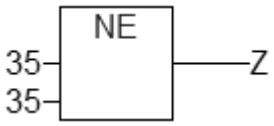



Рис. 10.24. Оператор порівняння **NE**

14. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.6. Вивчення блоків з математичними операторами**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **REAL**.
3. Створити **FBD**-схему (див. рис. 10.25).

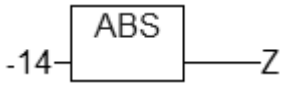



Рис. 10.25. Математичний оператор **ABS**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Створити **FBD**-схему (див. рис. 10.26).


Рис. 10.26. Математичний оператор **SQRT**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Створити **FBD**-схему (див. рис. 10.27).

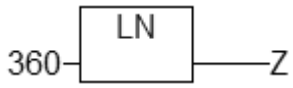



Рис. 10.27. Математичний оператор **LN**

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Створити **FBD**-схему (див. рис. 10.28).


Рис. 10.28. Математичний оператор **LOG**

10. В режимі виконання програмного коду визначити значення змінної **Z**.
11. Створити **FBD**-схему (див. рис. 10.29).


Рис. 10.29. Математичний оператор **EXP**

12. В режимі виконання програмного коду визначити значення змінної **Z**.
13. Створити **FBD**-схему (див. рис. 10.30).

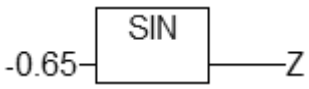



Рис. 10.30. Математичний оператор **SIN**

14. В режимі виконання програмного коду визначити значення змінної **Z**.
15. Створити **FBD**-схему (див. рис. 10.31).

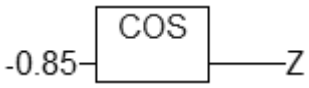



Рис. 10.31. Математичний оператор **COS**

16. В режимі виконання програмного коду визначити значення змінної **Z**.
17. Створити **FBD**-схему (див. рис. 10.32).


Рис. 10.32. Математичний оператор **TAN**

18. В режимі виконання програмного коду визначити значення змінної **Z**.
19. Створити **FBD**-схему (див. рис. 10.33).


Рис. 10.33. Математичний оператор **ASIN**

20. В режимі виконання програмного коду визначити значення змінної **Z**.
21. Створити **FBD**-схему (див. рис. 10.34).

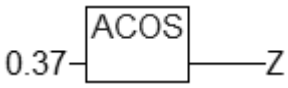



Рис. 10.34. Математичний оператор **ACOS**

22. В режимі виконання програмного коду визначити значення змінної **Z**.
23. Створити **FBD**-схему (див. рис. 10.35).


Рис. 10.35. Математичний оператор **ATAN**

24. В режимі виконання програмного коду визначити значення змінної **Z**.
25. Створити **FBD**-схему (див. рис. 10.36).

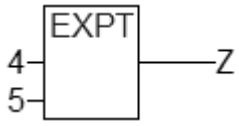



Рис. 10.36. Математичний оператор **EXPT**

26. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.7. Вивчення блоків з операторами перетворення типів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **DINT**.
3. Створити **FBD**-схему (див. рис. 10.37).

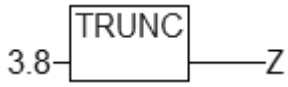



Рис. 10.37. Оператор перетворення типів **TRUNC**

4. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.8. Вивчення блоків з операторами присвоєння**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **REAL**.
3. Створити **FBD**-схему (див. рис. 10.38).


Рис. 10.38. Оператор присвоєння **MOVE**

4. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.2. Вивчення встановлення/скидання**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).

2. Створити змінну **Z** типу **BOOL** з початковим значенням **FALSE**.
3. Створити змінну **I** типу **INT** з початковим значенням **0**.
4. Створити змінну **L** типу **REAL**.
5. Створити **FBD**-схему (див. рис. 10.39).

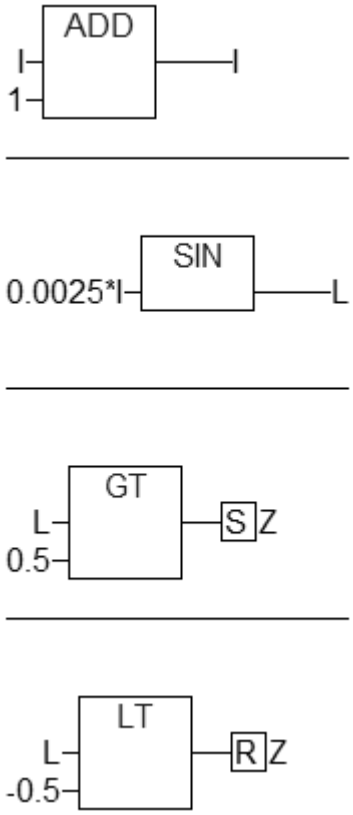



Рис. 10.39. Застосування встановлення/скидання

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.3. Вивчення інверсії логічних сигналів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити **FBD**-схему (див. рис. 10.40).

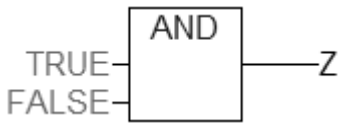



Рис. 10.40. Бітовий оператор **AND** без інверсії

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Виконати на **FBD**-схемі інверсію виходу бітового оператора **AND** (див. рис. 10.41).

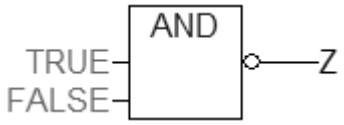



Рис. 10.41. Бітовий оператор **AND** з інверсією

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.4. Вивчення застосовування виразів на мові програмування ST**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **REAL**.
3. Створити **FBD**-схему (див. рис. 10.42).

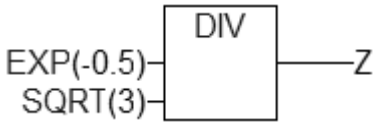



Рис. 10.42. Вирази на вході оператора **DIV**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **FBD**-схему (див. рис. 10.43).


Рис. 10.43. Вирази на вході оператора **MUL**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Змінити **FBD**-схему (див. рис. 10.44).

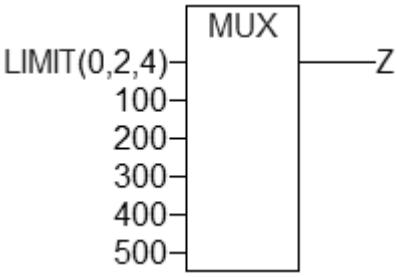



Рис. 10.44. Вирази на вході оператора **LIMIT**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5. Вивчення зворотних зв'язків**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).
2. Створити змінну **Z** типу **DINT** з початковим значенням **0**.
3. Створити **FBD**-схему (див. рис. 10.45).

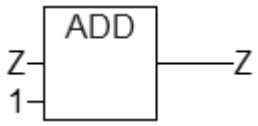



Рис. 10.45. Змінна **Z** в якості зворотного зв'язка

4. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.6. Вивчення коментарів**

1. Створити проект в середовищі програмування CoDeSys (для програми

**PLC_PRG** обрати мову програмування **FBD**).

2. Створити змінну **Z** типу **DINT**.

3. Створити **FBD**-схему (див. рис. 10.46).

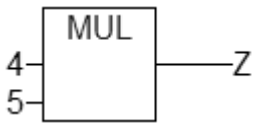



Рис. 10.46. **FBD**-схема без коментарів

4. В режимі виконання програмного коду визначити значення змінної **Z**.

5. Додати в **FBD**-схему коментар (див. рис. 10.47).

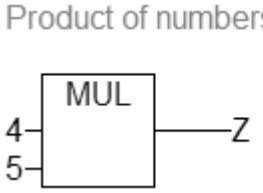



Рис. 10.47. **FBD**-схема з коментарями

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.7. Вивчення повернень**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).

2. Створити змінну **Z** типу **DINT** з початковим значенням **0**.

3. Створити змінну **I** типу **INT** з початковим значенням **0**.

4. Створити **FBD**-схему (див. рис. 10.48).

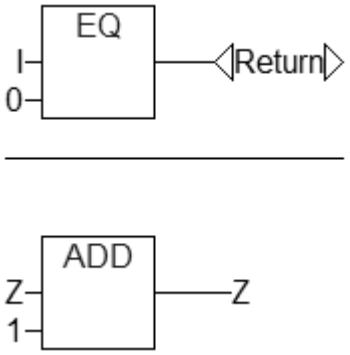



Рис. 10.48. Застосування повернень **Return**

5. В режимі виконання програмного коду визначити значення змінної **Z**.

6. Змінити початкове значення змінної **I** на **1**.

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.8. Вивчення міток і переходів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **FBD**).

2. Створити змінну **Z** типу **DINT** з початковим значенням **0**.

3. Створити змінну **I** типу **INT** з початковим значенням **0**.

4. Створити **FBD**-схему (див. рис. 10.49).

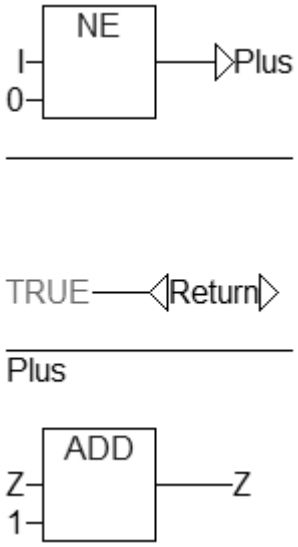



Рис. 10.49. Застосування міток і переходів

5. В режимі виконання програмного коду визначити значення змінної **Z**.
6. Змінити початкове значення змінної **I** на **1**.
7. В режимі виконання програмного коду визначити значення змінної **Z**.

### 3. Питання для самоперевірки

1. Що таке і які особливості має мова програмування **Function Block Diagram** (**FBD**)?
2. Як здійснюється відображення **POU** в мові програмування **Function Block Diagram** (**FBD**)?
3. Що таке з'єднувальні лінії в мові програмування **Function Block Diagram** (**FBD**)?
4. Який порядок виконання **FBD** в мові програмування **Function Block Diagram** (**FBD**)?
5. Як здійснюється інверсія логічних сигналів в мові програмування **Function Block Diagram** (**FBD**)?
6. Що таке з'єднувачі і зворотні зв'язки в мові програмування **Function Block Diagram** (**FBD**)?
7. Що таке мітки в мові програмування **Function Block Diagram** (**FBD**)?
8. Що таке переходи в мові програмування **Function Block Diagram** (**FBD**)?
9. Що таке повернення в мові програмування **Function Block Diagram** (**FBD**)?
10. Як застосовуються вирази **ST** в мові програмування **Function Block Diagram** (**FBD**)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №11.
# МОВА ПРОГРАМУВАННЯ LADDER DIAGRAM (LD)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення мови програмування **Ladder Diagram** (**LD**).

## 2. Порядок виконання лабораторної роботи

### 2.1. Вивчення замикаючих контактів

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити **LD**-схему (див. рис. 11.1).

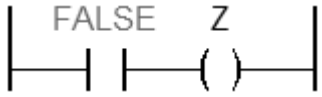



Рис. 11.1. Замикаючий контакт в розімкненому стані

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **LD**-схему (див. рис. 11.2).

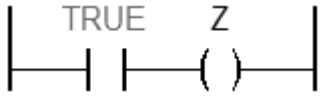



Рис. 11.2. Замикаючий контакт в замкненому стані

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Змінити **LD**-схему (див. рис. 11.3).


Рис. 11.3. Замикаючі контакти в розімкненому і замкненому станах

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Змінити **LD**-схему (див. рис. 11.4).


Рис. 11.4. Замикаючі контакти в замкненому стані

10. В режимі виконання програмного коду визначити значення змінної **Z**.

### 2.2. Вивчення розмикаючих контактів

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити **LD**-схему (див. рис. 11.5).

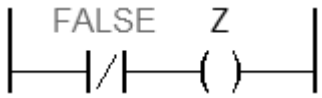



Рис. 11.5. Розмикаючий контакт в розімкненому стані

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **LD**-схему (див. рис. 11.6).

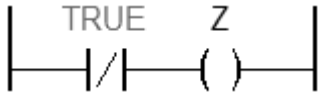



Рис. 11.6. Розмикаючий контакт в замкненому стані

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Змінити **LD**-схему (див. рис. 11.5).


Рис. 11.7. Розмикаючі контакти в розімкненому і замкненому станах

8. В режимі виконання програмного коду визначити значення змінної **Z**.
9. Змінити **LD**-схему (див. рис. 11.8).


Рис. 11.8. Розмикаючі контакти в розімкненому стані

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.3. Вивчення паралельних замикаючих контактів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити **LD**-схему (див. рис. 11.9).

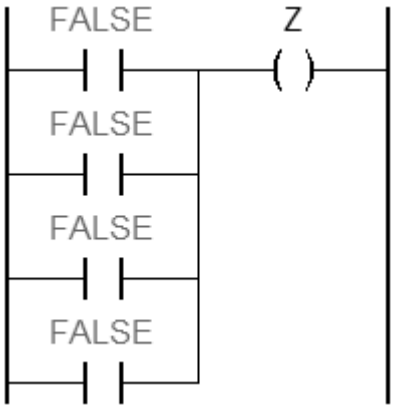



Рис. 11.9. Паралельні замикаючі контакти в розімкненому стані

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **LD**-схему (див. рис. 11.10).
6. В режимі виконання програмного коду визначити значення змінної **Z**.

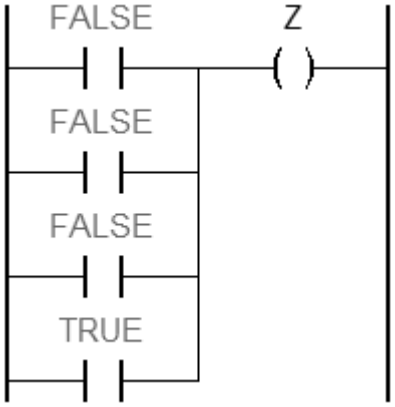



Рис. 11.10. Паралельні замикаючі контакти в розімкненому і замкненому станах

**2.4. Вивчення паралельних розмикаючих контактів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити **LD**-схему (див. рис. 11.11).

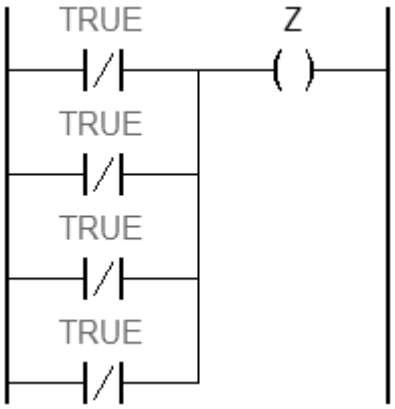



Рис. 11.11. Паралельні розмикаючі контакти в замкненому стані

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **LD**-схему (див. рис. 11.12).

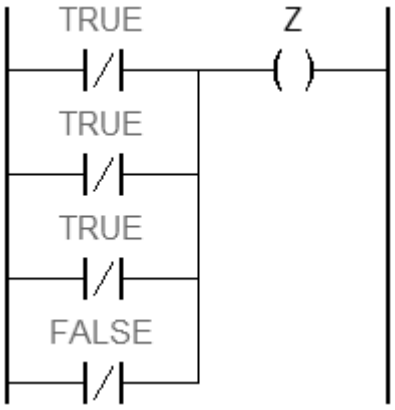



Рис. 11.12. Паралельні розмикаючі контакти в розімкненому і замкненому станах

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.5. Вивчення реле із встановленням/скиданням**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити змінну **X1** типу **BOOL** з початковим значенням **FALSE**.
4. Створити змінну **X2** типу **BOOL** з початковим значенням **FALSE**.
5. Створити **LD**-схему (див. рис. 11.11).

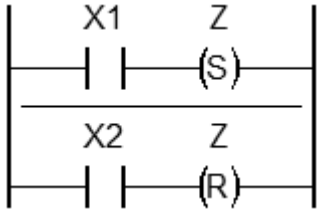



Рис. 11.13. Реле із встановленням/скиданням

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X1** на **TRUE** і визначити значення змінної **Z**.
8. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X1** на **FALSE** і визначити значення змінної **Z**.
9. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X2** на **TRUE** і визначити значення змінної **Z**.
10. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X2** на **FALSE** і визначити значення змінної **Z**.

**2.6. Вивчення зворотних зв’язків**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **BOOL**.
3. Створити змінну **X1** типу **BOOL** з початковим значенням **FALSE**.
4. Створити змінну **X2** типу **BOOL** з початковим значенням **FALSE**.
5. Створити **LD**-схему (див. рис. 11.14).

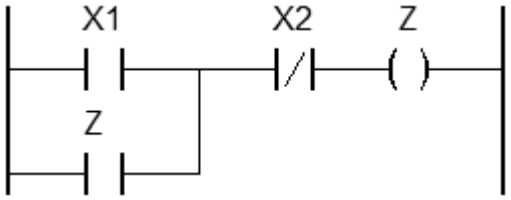



Рис. 11.14. Зворотний зв’язок

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X1** на **TRUE** і визначити значення змінної **Z**.
8. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X1** на **FALSE** і визначити значення змінної **Z**.
9. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X2** на **TRUE** і визначити значення змінної **Z**.

10. Не виходячи з режиму виконання програмного коду, змінити значення змінної **X2** на **FALSE** і визначити значення змінної **Z**.

**2.7. Вивчення застосовування блоків на мові програмування FBD**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **DINT** з початковим значенням **0**.
3. Створити змінну **X** типу **BOOL**.
4. Створити змінну **L** типу **BOOL**.
5. Створити екземпляр **_CTU_** функціонального блока **CTU**.
6. Створити **LD**-схему (див. рис. 11.15).

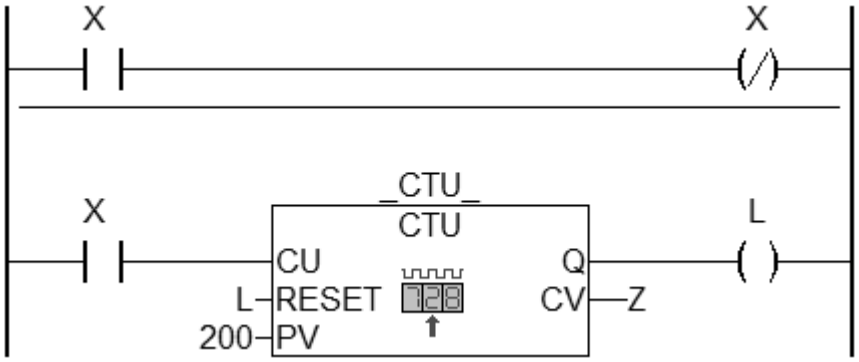



Рис. 11.15. Застосовування блоків на мові програмування **FBD**

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.8. Вивчення застосовування блоків на мові програмування FBD з входом EN**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **DINT**.
3. Створити **LD**-схему (див. рис. 11.16).

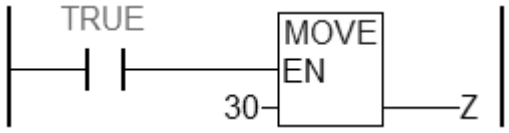



Рис. 11.16. Застосовування блоків на мові програмування **FBD**, на вхід **EN** яких подано **TRUE**

4. В режимі виконання програмного коду визначити значення змінної **Z**.
5. Змінити **LD**-схему (див. рис. 11.17).

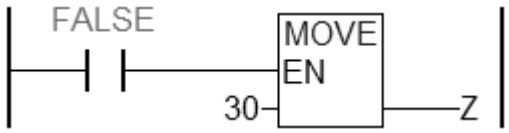



Рис. 11.17. Застосовування блоків на мові програмування **FBD**, на вхід **EN** яких подано **FALSE**

6. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.9. Вивчення повернень**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **DINT** з початковим значенням **0**.
3. Створити змінну **I** типу **INT** з початковим значенням **0**.
4. Створити змінну **L** типу **BOOL**.
5. Створити **LD**-схему (див. рис. 11.18).

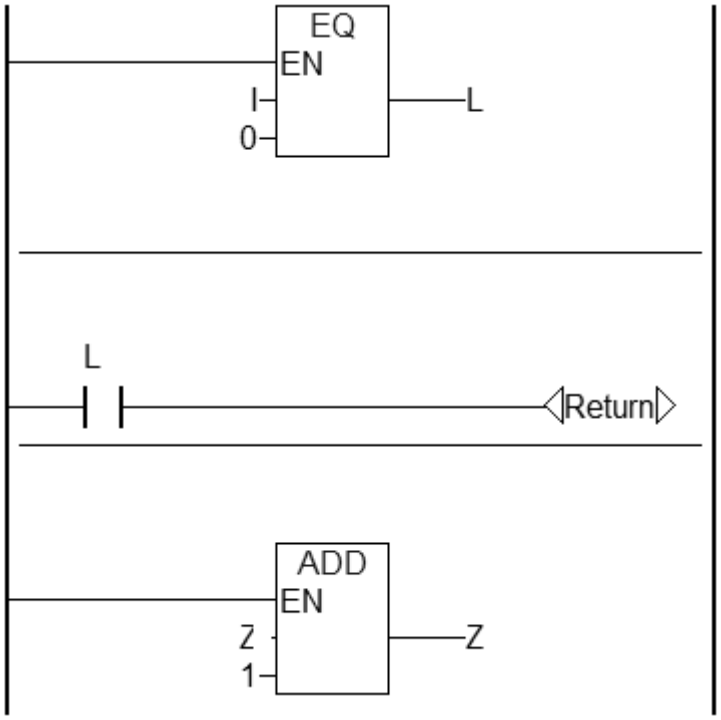
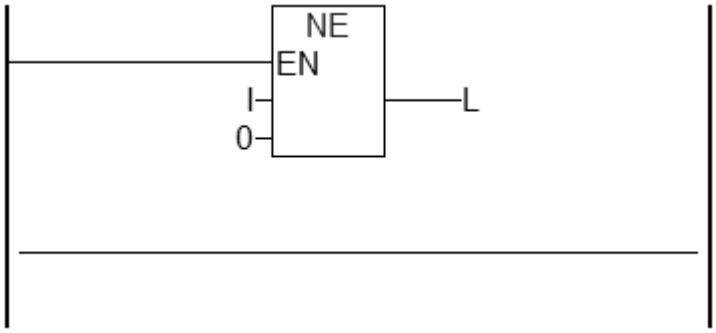



Рис. 11.18. Застосування повернень **Return**

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Змінити значення змінної **I** на **1**.
8. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.10. Вивчення міток і переходів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **LD**).
2. Створити змінну **Z** типу **DINT** з початковим значенням **0**.
3. Створити змінну **I** типу **INT** з початковим значенням **0**.
4. Створити змінну **L** типу **BOOL**.
5. Створити **LD**-схему (див. рис. 11.19).



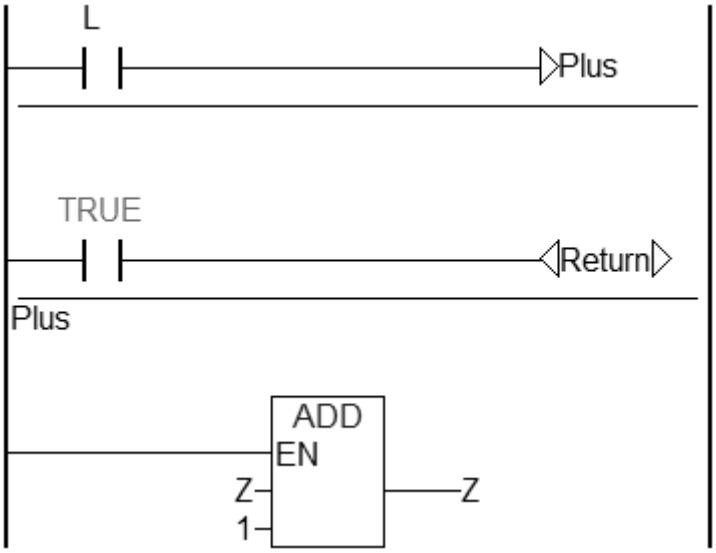



Рис. 11.19. Застосування міток і переходів

6. В режимі виконання програмного коду визначити значення змінної **Z**.
7. Змінити початкове значення змінної **I** на **1**.
8. В режимі виконання програмного коду визначити значення змінної **Z**.

### 3. Питання для самоперевірки

1. Що таке і які особливості має мова програмування **Ladder Diagram** (**LD**)?
2. Що таке ланцюги в мові програмування **Ladder Diagram** (**LD**)?
3. Що таке реле із встановленням в мові програмування **Ladder Diagram** (**LD**)?
4. Що таке реле із скиданням в мові програмування **Ladder Diagram** (**LD**)?
5. Що таке порядок виконання в мові програмування **Ladder Diagram** (**LD**)?
6. Що таке зворотні зв'язки в мові програмування **Ladder Diagram** (**LD**)?
7. Як здійснюється керування порядком виконання в мові програмування **Ladder Diagram** (**LD**)?
8. Як в **LD**-схему можна вбудовувати функції на мові програмування **Function Block Diagram** (**FBD**)?
9. Як в **LD**-схему можна вбудовувати функціональні блоки на мові програмування **Function Block Diagram** (**FBD**)?
10. Які існують особливості реалізації мови програмування **Ladder Diagram** (**LD**) в CoDeSys?
11. Які існують особливості **LD**-схем в режимі виконання в мові програмування **Ladder Diagram** (**LD**)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №12.
# МОВА ПРОГРАМУВАННЯ CONTINUOUS FUNCTION CHART (CFC)

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення мови програмування **Continuous Function Chart** (**CFC**).

## 2. Порядок виконання лабораторної роботи

### 2.1. Вивчення блоків

#### 2.1.1. Вивчення арифметичних операторів

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **2**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **3**.

4. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **DINT**.

7. З'єднати вихід *Блока* **ADD** з входом *Виходу блока* **Output** (див. рис. 12.1).

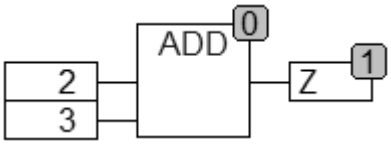



Рис. 12.1. Арифметичний оператор **ADD**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Виконати аналогічні дії при інших значеннях констант всередині першого *Входу блока* **Input**, другого *Входу блока* **Input** і *Блоками* **SUB**, **MUL**, **DIV** і **MOD**.

#### 2.1.2. Вивчення бітових операторів

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **2#1001_0011**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **2#1000_1010**.

4. Перенести в робочу область **CFC**-редактора *Блок* **AND**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **AND**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмі-

стити всередині *Виходу блока* **Output** змінну **Z** типу **BYTE**.

7. З'єднати вихід *Блока* **AND** з входом *Виходу блока* **Output** (див. рис. 12.2).

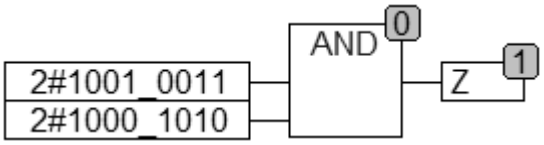



Рис. 12.2. Бітовий оператор **AND**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Виконати аналогічні дії при інших значеннях констант всередині першого *Входу блока* **Input**, другого *Входу блока* **Input** і *Блоками* **OR**, **XOR** і **NOT**.

**2.1.3. Вивчення операторів зсуву**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **16#54**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **2**.

4. Перенести в робочу область **CFC**-редактора *Блок* **SHL**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **SHL**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **BYTE**.

7. З'єднати вихід *Блока* **SHL** з входом *Виходу блока* **Output** (див. рис. 12.3).

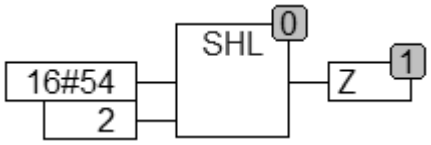



Рис. 12.3. Оператор зсуву **SHL**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Виконати аналогічні дії при інших значеннях констант всередині першого *Входу блока* **Input**, другого *Входу блока* **Input** і *Блоками* **SHR**, **ROL** і **ROR**.

**2.1.4. Вивчення операторів вибірки**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **TRUE**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **10**.

4. Перенести в робочу область **CFC**-редактора третій *Вхід блока* **Input**. Розмістити всередині третього *Входу блока* **Input** константу **20**.

5. Перенести в робочу область **CFC**-редактора *Блок* **SEL**.

6. З'єднати виходи першого, другого і третього *Входів блоків* **Input** з відповідними входами *Блока* **SEL**.

7. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **DINT**.

8. З'єднати вихід *Блока* **SEL** з входом *Виходу блока* **Output** (див. рис. 12.4).

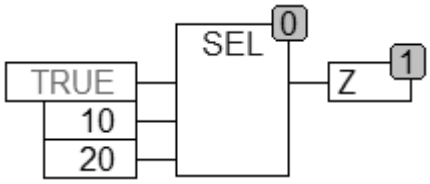



Рис. 12.4. Оператор вибірки **SEL**

9. В режимі виконання програмного коду визначити значення змінної **Z**.

10. Виконати аналогічні дії при інших значеннях констант всередині першого *Входу блока* **Input**, другого *Входу блока* **Input** (і за необхідності інших *Входів блоків* **Input**) і *Блоками* **MAX**, **MIN**, **LIMIT** і **MUX**.

**2.1.5. Вивчення операторів порівняння**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **25**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **50**.

4. Перенести в робочу область **CFC**-редактора *Блок* **GT**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **GT**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **BOOL**.

7. З'єднати вихід *Блока* **GT** з входом *Виходу блока* **Output** (див. рис. 12.5).

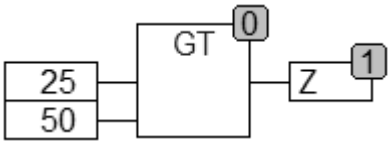



Рис. 12.5. Оператор порівняння **GT**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Виконати аналогічні дії при інших значеннях констант всередині першого *Входу блока* **Input**, другого *Входу блока* **Input** і *Блоками* **LT**, **GE**, **LE**, **EQ** і **NE**.

**2.1.6. Вивчення математичних операторів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **–15**.

3. Перенести в робочу область **CFC**-редактора *Блок* **ABS**.

4. З'єднати вихід *Входу блока* **Input** з відповідним входом *Блока* **ABS**.

5. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **REAL**.

6. З'єднати вихід *Блока* **ABS** з входом *Виходу блока* **Output** (див. рис. 12.2).

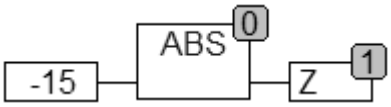



Рис. 12.6. Математичний оператор **ABS**

7. В режимі виконання програмного коду визначити значення змінної **Z**.

8. Виконати аналогічні дії при інших значеннях констант всередині *Входу блока* **Input** (і за необхідності інших *Входів блоків* **Input**) і *Блоками* **SQRT**, **LN**, **LOG**, **EXP**, **SIN**, **COS**, **TAN**, **ASIN**, **ACOS**, **ATAN** і **EXPT**.

**2.1.7. Вивчення операторів перетворення типів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **2.7**.

3. Перенести в робочу область **CFC**-редактора *Блок* **TRUNC**.

4. З'єднати вихід *Входу блока* **Input** з відповідним входом *Блока* **TRUNC**.

5. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **REAL**.

6. З'єднати вихід *Блока* **TRUNC** з входом *Виходу блока* **Output** (див. рис. 12.7).

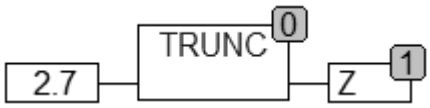



Рис. 12.7. Оператор перетворення типів **TRUNC**

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.1.8. Вивчення операторів присвоєння**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **6.4**.

3. Перенести в робочу область **CFC**-редактора *Блок* **MOVE**.

4. З'єднати вихід *Входу блока* **Input** з відповідним входом *Блока* **MOVE**.

5. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **REAL**.

6. З'єднати вихід *Блока* **MOVE** з входом *Виходу блока* **Output** (див. рис. 12.7).

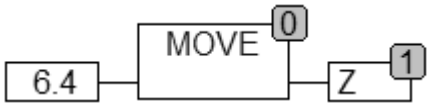



Рис. 12.8. Оператор присвоєння **MOVE**

7. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.2. Вивчення повернень**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** змінну **I** з початковим значенням **0**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **0**.

4. Перенести в робочу область **CFC**-редактора *Блок* **EQ**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **EQ**.

6. Перенести в робочу область **CFC**-редактора перший *Вихід блока* **Output**. Розмістити всередині першого *Виходу блока* **Output** *Повернення* **Return**.

7. З'єднати *Вихід блока* **EQ** з входом першого *Виходу блока* **Output**.

8. Перенести в робочу область **CFC**-редактора третій *Вхід блока* **Input**. Розмістити всередині третього *Входу блока* **Input** змінну **Z** з початковим значенням **0**.

9. Перенести в робочу область **CFC**-редактора четвертий *Вхід блока* **Input**. Розмістити всередині четвертого *Входу блока* **Input** константу **1**.

10. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

11. З'єднати виходи третього і четвертого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

12. Перенести в робочу область **CFC**-редактора другий *Вихід блока* **Output**. Розмістити всередині другого *Виходу блока* **Output** змінну **Z**.

13. З'єднати вихід *Блока* **ADD** з входом другого *Виходу блока* **Output** (див. рис. 12.9).

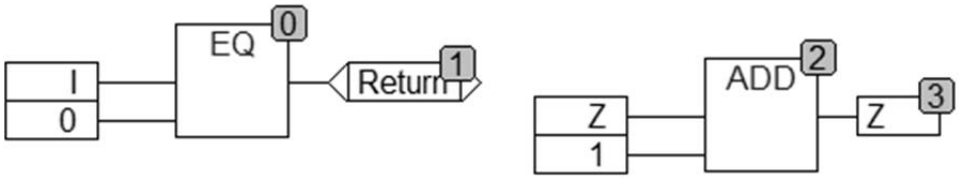



Рис. 12.9. Застосування повернень

14. В режимі виконання програмного коду визначити значення змінної **Z**.

15. Змінити з початкове значення змінної **I** на **1**.

16. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.3. Вивчення міток і переходів**

1. Створити проект в середовищі програмування CoDeSys (для програми

**PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** змінну **I** з початковим значенням **0**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **0**.

4. Перенести в робочу область **CFC**-редактора *Блок* **NE**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **NE**.

6. Перенести в робочу область **CFC**-редактора перший *Вихід блока* **Output**. Розмістити всередині першого *Виходу блока* **Output** *Перехід* **Plus**.

7. З'єднати вихід *Блока* **NE** з входом першого *Виходу блока* **Output**.

8. Перенести в робочу область **CFC**-редактора третій *Вхід блока* **Input**. Розмістити всередині третього *Входу блока* **Input** константу **TRUE**.

9. Перенести в робочу область **CFC**-редактора *Повернення* **Return**.

10. З'єднати вихід третього *Входу блока* **Input** з входом *Повернення* **Return**.

11. Перенести в робочу область **CFC**-редактора *Мітку* **Plus**.

12. Перенести в робочу область **CFC**-редактора четвертий *Вхід блока* **Input**. Розмістити всередині четвертого *Входу блока* **Input** змінну **Z**.

13. Перенести в робочу область **CFC**-редактора п'ятий *Вхід блока* **Input**. Розмістити всередині п'ятого *Входу блока* **Input** константу **1**.

14. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

15. З'єднати виходи четвертого і п'ятого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

16. Перенести в робочу область **CFC**-редактора другий *Вихід блока* **Output**. Розмістити всередині другого *Виходу блока* **Output** змінну **Z**.

17. З'єднати вихід *Блока* **ADD** з входом другого *Виходу блока* **Output** (див. рис. 12.10).

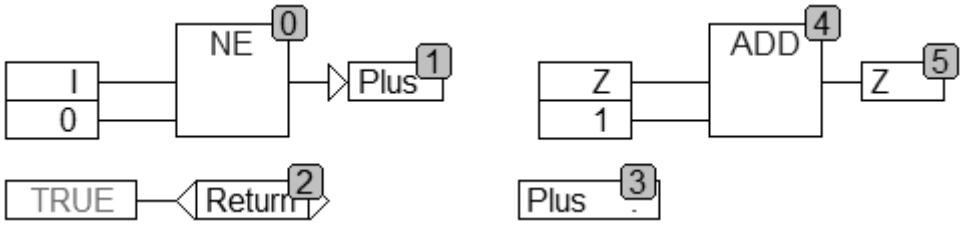



Рис. 12.10. Застосування міток і переходів

18. В режимі виконання програмного коду визначити значення змінної **Z**.

19. Змінити з початкове значення змінної **I** на **1**.

20. В режимі виконання програмного коду визначити значення змінної **Z**.

### 2.4. Вивчення коментарів

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора *Коментар* **Comment**. Розмістити всередині *Коментаря* **Comment** певний текст (див. рис. 12.11).


Рис. 12.11. Застосування коментарів

3. Виконати аналогічні дії при інших значеннях тексту всередині *Коментаря* **Comment**.

**2.5. Вивчення інверсії**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **2#1001_0011**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **2#1000_1010**.

4. Перенести в робочу область **CFC**-редактора *Блок* **AND**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **AND**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **BOOL**.

7. З'єднати *Вихід блока* **AND** з входом *Виходу блока* **Output**.

8. Застосувати *Інверсію* до виходу *Блока* **AND** (див. рис. 12.12).

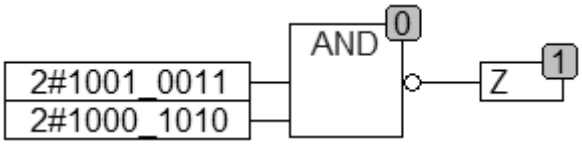



Рис. 12.12. Застосування інверсії

9. В режимі виконання програмного коду визначити значення змінної **Z**.

10. Виконати аналогічні дії при інших значеннях констант всередині першого *Входу блока* **Input**, другого *Входу блока* **Input** і *Блоками* **OR**, **XOR** і **NOT**.

**2.6. Вивчення дозволу входу/дозволу виходу**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **TRUE**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **60**.

4. Перенести в робочу область **CFC**-редактора *Блок* **MOVE**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **MOVE**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **DINT**.

7. З'єднати вихід *Блока* **MOVE** з входом *Виходу блока* **Output** (див. рис. 12.4).

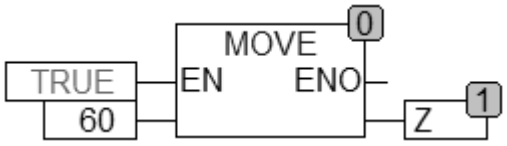



Рис. 12.13. Застосовування блоків на мові програмування **CFC**, на вхід **EN** яких подано **TRUE**

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Розмістити всередині першого Входу блока **Input** константу **TRUE** (див. рис. 12.5).

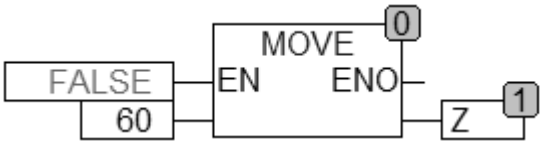



Рис. 12.14. Застосовування блоків на мові програмування **CFC**, на вхід **EN** яких подано **FALSE**

10. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.7. Вивчення встановлення/скидання**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** змінну **I** типу **INT** з початковим значенням **1**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **1**.

4. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

6. Перенести в робочу область **CFC**-редактора перший *Вихід блока* **Output**. Розмістити всередині першого *Виходу блока* **Output** змінну **I**.

7. З'єднати вихід *Блока* **ADD** з входом першого *Виходу блока* **Output**.

8. Перенести в робочу область **CFC**-редактора третій *Вхід блока* **Input**. Розмістити всередині третього *Входу блока* **Input** змінну **I**.

9. Перенести в робочу область **CFC**-редактора четвертий *Вхід блока* **Input**. Розмістити всередині четвертого *Входу блока* **Input** константу **0.0025**.

10. Перенести в робочу область **CFC**-редактора *Блок* **MUL**.

11. З'єднати виходи третього і четвертого *Входів блоків* **Input** з відповідними входами *Блока* **MUL**.

12. Перенести в робочу область **CFC**-редактора *Блок* **SIN**.

13. З'єднати вихід *Блока* **MUL** з входом *Блока* **SIN**.

14. Перенести в робочу область **CFC**-редактора другий *Вихід блока* **Output**. Розмістити всередині другого *Виходу блока* **Output** змінну **L** типу **REAL**.

15. З'єднати вихід *Блока* **SIN** з входом другого *Виходу блока* **Output**.

16. Перенести в робочу область **CFC**-редактора п’ятий *Вхід блока* **Input**. Розмістити всередині п’ятого *Входу блока* **Input** змінну **L**.

17. Перенести в робочу область **CFC**-редактора шостий *Вхід блока* **Input**. Розмістити всередині шостого *Входу блока* **Input** константу **0.5**.

18. Перенести в робочу область **CFC**-редактора *Блок* **GT**.

19. З’єднати виходи п’ятого і шостого *Входів блоків* **Input** з відповідними входами *Блока* **GT**.

20. Перенести в робочу область **CFC**-редактора третій *Вихід блока* **Output**. Розмістити всередині третього *Виходу блока* **Output** змінну **Z** типу **BOOL**.

21. Застосувати до змінної **Z** *Встановлення* **S**.

22. З’єднати вихід *Блока* **ADD** з входом третього *Виходу блока* **Output**.

23. Перенести в робочу область **CFC**-редактора сьомий *Вхід блока* **Input**. Розмістити всередині сьомого *Входу блока* **Input** змінну **L**.

24. Перенести в робочу область **CFC**-редактора восьмий *Вхід блока* **Input**. Розмістити всередині восьмого *Входу блока* **Input** константу **–0.5**.

25. Перенести в робочу область **CFC**-редактора *Блок* **LT**.

26. З’єднати виходи сьомого і восьмого *Входів блоків* **Input** з відповідними входами *Блока* **LT**.

27. Перенести в робочу область **CFC**-редактора четвертий *Вихід блока* **Output**. Розмістити всередині четвертого *Виходу блока* **Output** змінну **Z**.

28. Застосувати до змінної **Z** *Скидання* **R**.

29. З’єднати вихід *Блока* **LT** з входом четвертого *Виходу блока* **Output** (див. рис. 12.15).

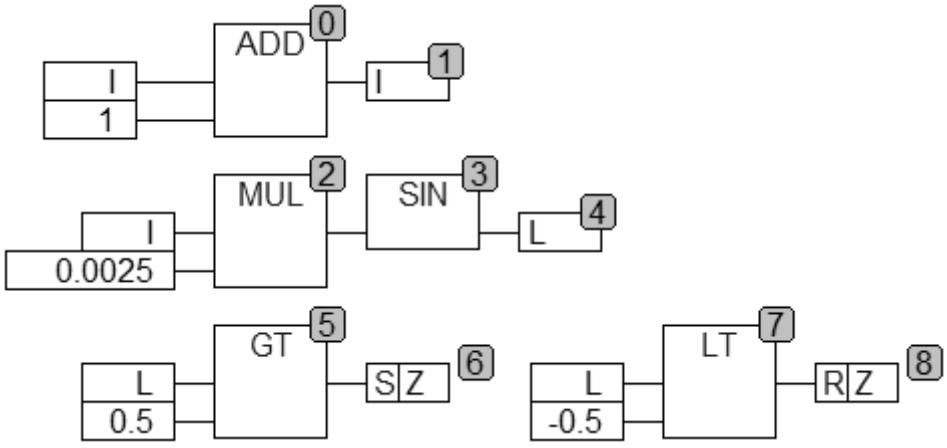



Рис. 12.15. Застосування встановлення/скидання

30. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.8. Вивчення макрокоманд**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **5**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **10**.

4. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

5. З'єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмістити всередині *Виходу блока* **Output** змінну **Z** типу **DINT**.

7. З'єднати вихід *Блока* **ADD** з входом *Виходу блока* **Output** (див. рис. 12.16).

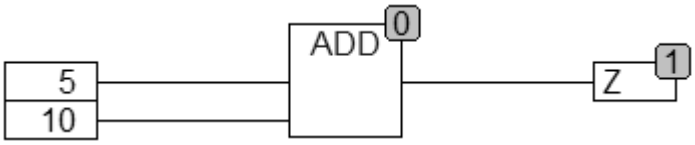



Рис. 12.16. **CFC**-схема без макрокоманд (варіант 1)

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Застосувати до *Блока* **ADD** *Макрокоманду* (див. рис. 12.17).

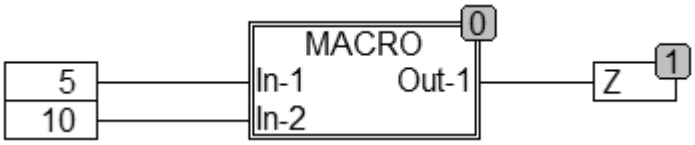



Рис. 12.17. **CFC**-схема з макрокомандами (варіант 2)

10. В режимі виконання програмного коду визначити значення змінної **Z**.

11. Застосувати до першого і другого *Входів блоків* **Input** та *Блока* **ADD** *Макрокоманду* (див. рис. 12.18).


Рис. 12.18. **CFC**-схема з макрокомандами (варіант 3)

12. В режимі виконання програмного коду визначити значення змінної **Z**.

13. Застосувати до *Блока* **ADD** і *Виходу блока* **Output** *Макрокоманду* (див. рис. 12.19).

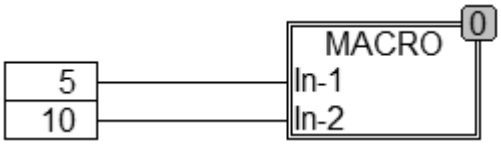



Рис. 12.19. **CFC**-схема з макрокомандами (варіант 4)

14. В режимі виконання програмного коду визначити значення змінної **Z**.

15. Застосувати до всієї **CFC**-схеми *Макрокоманду* (див. рис. 12.20).

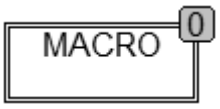



Рис. 12.20. **CFC**-схема з макрокомандами (варіант 5)

16. В режимі виконання програмного коду визначити значення змінної **Z**.

**2.9. Вивчення порядку виконання CFC-схеми**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **10**.

3. Перенести в робочу область **CFC**-редактора перший *Блок* **MOVE**.

4. З'єднати вихід першого *Входу блока* **Input** з входом першого *Блока* **MOVE**.

5. Перенести в робочу область **CFC**-редактора перший *Вихід блока* **Output**. Розмістити всередині першого *Виходу блока* **Output** змінну **Z1** типу **DINT**.

6. З'єднати вихід першого *Блока* **MOVE** з входом першого *Виходу блока* **Output**.

7. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **10**.

8. Перенести в робочу область **CFC**-редактора другий *Блок* **MOVE**.

9. З'єднати вихід другого *Входу блока* **Input** з входом другого *Блока* **MOVE**.

10. Перенести в робочу область **CFC**-редактора другий *Вихід блока* **Output**. Розмістити всередині другого *Виходу блока* **Output** змінну **Z2** типу **DINT**.

11. З'єднати вихід другого *Блока* **MOVE** з входом другого *Виходу блока* **Output**.

12. Перенести в робочу область **CFC**-редактора третій *Вхід блока* **Input**. Розмістити всередині третього *Входу блока* **Input** змінну **Z2**.

13. Перенести в робочу область **CFC**-редактора четвертий *Вхід блока* **Input**. Розмістити всередині четвертого *Входу блока* **Input** константу **35**.

14. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

15. З'єднати виходи третього і четвертого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

16. Перенести в робочу область **CFC**-редактора третій *Вихід блока* **Output**. Розмістити всередині третього *Виходу блока* **Output** змінну **Z1**.

17. З'єднати вихід *Блока* **ADD** з входом третього *Виходу блока* **Output**.

18. Перенести в робочу область **CFC**-редактора п'ятий *Вхід блока* **Input**. Розмістити всередині п'ятого *Входу блока* **Input** змінну **Z1**.

19. Перенести в робочу область **CFC**-редактора шостий *Вхід блока* **Input**. Розмістити всередині шостого *Входу блока* **Input** константу **15**.

20. Перенести в робочу область **CFC**-редактора *Блок* **SUB**.

21. З'єднати виходи п'ятого і шостого *Входів блоків* **Input** з відповідними входами *Блока* **SUB**.

22. Перенести в робочу область **CFC**-редактора четвертий *Вихід блока* **Output**. Розмістити всередині четвертого *Виходу блока* **Output** змінну **Z2**.

23. З'єднати вихід *Блока* **SUB** з входом четвертого *Виходу блока* **Output** (див. рис. 12.21).

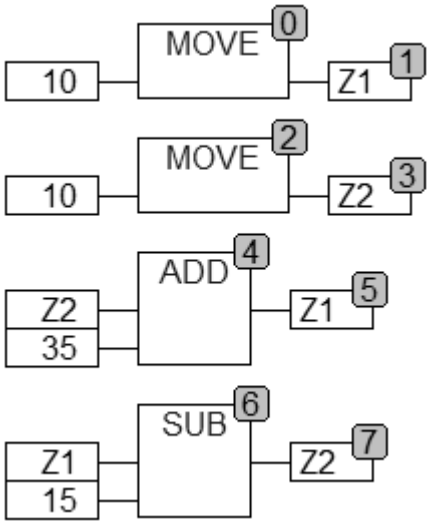



Рис. 12.21. **CFC**-схема (порядок виконання згідно з “потоком” даних)

24. В режимі виконання програмного коду визначити значення змінних **Z1** і **Z2**.

25. Змінити порядок виконання програмного коду (див. рис. 12.22).

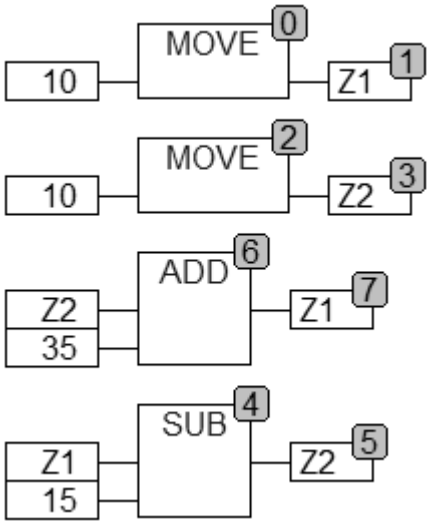



Рис. 12.22. **CFC**-схема (порядок виконання довільний)

26. В режимі виконання програмного коду визначити значення змінних **Z1** і **Z2**.

**2.10. Вивчення з’єднуючих маркерів**

1. Створити проект в середовищі програмування CoDeSys (для програми **PLC_PRG** обрати мову програмування **CFC**).

2. Перенести в робочу область **CFC**-редактора перший *Вхід блока* **Input**. Розмістити всередині першого *Входу блока* **Input** константу **5**.

3. Перенести в робочу область **CFC**-редактора другий *Вхід блока* **Input**. Розмістити всередині другого *Входу блока* **Input** константу **10**.

4. Перенести в робочу область **CFC**-редактора *Блок* **ADD**.

5. З’єднати виходи першого і другого *Входів блоків* **Input** з відповідними входами *Блока* **ADD**.

6. Перенести в робочу область **CFC**-редактора *Вихід блока* **Output**. Розмі-

стити всередині *Виходу блока* **Output** змінну **Z** типу **DINT**.

7. З'єднати вихід *Блока* **ADD** з входом *Виходу блока* **Output** (див. рис. 12.23).

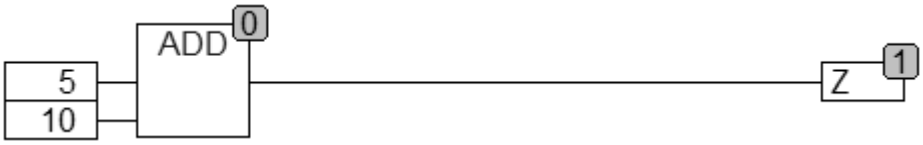



Рис. 12.23. **CFC**-схема без з'єднуючих маркерів

8. В режимі виконання програмного коду визначити значення змінної **Z**.

9. Застосувати до *Блока* **ADD** з'єднуючий маркер (див. рис. 12.24).


Рис. 12.24. **CFC**-схема з з'єднуючими маркерами

10. В режимі виконання програмного коду визначити значення змінної **Z**.

### 3. Питання для самоперевірки

1. Що таке і які особливості має мова програмування **Continuous Function Chart** (**CFC**)?
2. Що таке входи блоків і виходи блоків в мові програмування **Continuous Function Chart** (**CFC**)?
3. Що таке блоки в мові програмування **Continuous Function Chart** (**CFC**)?
4. Що таке переходи, мітки і повернення в мові програмування **Continuous Function Chart** (**CFC**)?
5. Як задаються коментарі в мові програмування **Continuous Function Chart** (**CFC**)?
6. Що таке інверсія в мові програмування **Continuous Function Chart** (**CFC**)?
7. Що таке встановлення/скидання в мові програмування **Continuous Function Chart** (**CFC**)?
8. Що таке дозвіл входу і дозвіл виходу в мові програмування **Continuous Function Chart** (**CFC**)?
9. Що таке макрокоманди, входи макрокоманди і виходи макрокоманди в мові програмування **Continuous Function Chart** (**CFC**)?
10. Який порядок виконання **CFC**-схеми в мові програмування **Continuous Function Chart** (**CFC**)?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №13.
# ВИВЧЕННЯ РОБОТИ АНАЛОГОВИХ ВХОДІВ ПЛК150 ВИРОБНИЦТВА ТОВ “ВО ОВЕН”

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи аналогових входів ПЛК150 виробництва ТОВ “ВО ОВЕН”.

## 2. Порядок виконання лабораторної роботи

### 2.1. Вимірювання температури за допомогою термоелектричного перетворювача

1. Підключити до аналогового входу **AI–1–0** термоелектричний перетворювач типу ТХК(L), ТХА(K), ТЖК (J), ТПР (B) тощо.

2. В лівій частині вікна редактора *Конфігуратора ПЛК* для аналогового входу **AI–1–0** обрати модуль аналогового входу **Thermocouple sensor [Slot]**.

3. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Name** для параметра **Type of sensor** обрати значення, яке відповідає термоелектричному перетворювачу з відповідною номінальною статичною характеристикою.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Measure interval, s** обрати період часу між двома послідовними вимірюваннями.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **First point** (за необхідності) обрати значення температури, для якої буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

6. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Second point** (за необхідності) обрати значення температури, для якої буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

7. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Third point** (за необхідності) обрати значення температури, для якої буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Thermocouple sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta**

обрати значення, на яке буде виконане коригування результату вимірювання.

8. Виконати вимірювання температури термоелектричним перетворювачем типу ТХК(L), ТХА(K), ТЖК (J), ТПР (B) тощо (при цьому для температури необхідно задати кілька різних значень). Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2. Вимірювання температури за допомогою термометра опору**

1. Підключити до аналогового входу **AI–2–0** термометр опору типу ТСМ, ТСП, ТСН, Pt тощо.

2. В лівій частині вікна редактора *Конфігуратора ПЛК* для аналогового входу **AI–2–0** обрати модуль аналогового входу **RTD sensor [Slot]**.

3. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Name** для параметра **Type of sensor** обрати значення, яке відповідає термометру опору з відповідною номінальною статичною характеристикою.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Measure interval, s** обрати період часу між двома послідовними вимірюваннями.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **First point** (за необхідності) обрати значення температури, для якої буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

6. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Second point** (за необхідності) обрати значення температури, для якої буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

7. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Third point** (за необхідності) обрати значення температури, для якої буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **RTD sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

8. Виконати вимірювання температури термометром опору ТСМ, ТСП, ТСН, Pt тощо (при цьому для температури необхідно задати кілька різних значень). Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**3. Вимірювання уніфікованих електричних сигналів**

1. Підключити до аналогового входу **AI–3–0** уніфікований електричний сигнал (в вигляді напруги або в вигляді струму).

2. В лівій частині вікна редактора *Конфігуратора ПЛК* для аналогового входу **AI–3–0** обрати модуль аналогового входу **Unified sensor [Slot]**.

3. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]** на вкладці **Module parameters** в полі **Name** для параметра **Type of sensor** обрати значення, яке відповідає відповідному уніфікованому електричному сигналу.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]**на вкладці **Module parameters** в полі **Value** для параметра **Measure interval, s** обрати період часу між двома послідовними вимірюваннями.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **First point** (за необхідності) обрати значення уніфікованого електричного сигналу, для якого буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]**на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

6. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Second point** (за необхідності) обрати значення уніфікованого електричного сигналу, для якого буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

7. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Third point** (за необхідності) обрати значення уніфікованого електричного сигналу, для якого буде виконане коригування результату вимірювання. Відповідно до цього, в правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Unified sensor [Slot]**на вкладці **Module parameters** в полі **Value** для параметра **Delta** обрати значення, на яке буде виконане коригування результату вимірювання.

8. Виконати вимірювання уніфікованого електричного сигналу. Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

9. Зіставити уніфікованому електричному сигналу значення певного технологічного параметра (наприклад, тиску, відносної вологості, концентрації тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**3.1. Визначення стану контактного перетворювача**

1. Підключити до аналогового входу **AI–4–0** механічний або електронний контактний перетворювач.

2. В лівій частині вікна редактора *Конфігуратора ПЛК* для аналогового входу **AI–4–0** обрати модуль аналогового входу **Contact sensor [Slot]**.

3. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Contact sensor [Slot]** на вкладці **Module parameters** в полі **Name** для параметра **Type of sensor** обрати значення **KT**.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Contact sensor [Slot]** на вкладці **Module parameters** в полі **Value** для параметра **Measure interval, s** обрати період часу між двома послідовними вимірюваннями.

5. Виконати вимірювання стану механічного або електронного контактного перетворювача. Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**4. Питання для самоперевірки**

1. Як можна підключити до ПЛК150 такі вимірювальні перетворювачі температури, як термометри опору?

2. Як можна підключити до ПЛК150 такі вимірювальні перетворювачі температури, як термоелектричні перетворювачі?

3. Як можна підключити до ПЛК150 джерела уніфікованої електричної напруги?

4. Як можна підключити до ПЛК150 джерела уніфікованого електричного струму?

5. Як можна підключити до ПЛК150 контактні перетворювачі?

6. Як у вікні редактора конфігуратора ПЛК можна обрати модуль аналогового входу?

7. Як у вікні редактора конфігуратора ПЛК можна обрати номінальну статичну характеристику для термоелектричного перетворювача або термометра опору?

8. Як у вікні редактора конфігуратора ПЛК можна обрати значення, яке відповідає уніфікованій електричній напрузі або уніфікованому електричному струму?

9. Як у вікні редактора конфігуратора ПЛК можна обрати значення, яке відповідає контактному перетворювачу?

10. Як у вікні редактора конфігуратора ПЛК можна обрати період часу між двома послідовними вимірюваннями?

11. Як у вікні редактора конфігуратора ПЛК можна виконати коригування результату вимірювання?

**5. Рекомендована література**

1. Основна література [1о–8о].

2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №14.
# ВИВЧЕННЯ РОБОТИ ДИСКРЕТНИХ ВХОДІВ ПЛК150 ВИРОБНИЦТВА ТОВ “ВО ОВЕН”

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи дискретних входів ПЛК150 виробництва ТОВ “ВО ОВЕН”.

## 2. Порядок виконання лабораторної роботи

### 2.1. Визначення стану контактного перетворювача

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Time of filtration** обрати період часу між двома послідовними вимірюваннями (задається в сотнях мікросекунд – наприклад, значення 5 параметра **Time of filtration** відповідає періоду часу між двома послідовними вимірюваннями 500 мкс ).

3. Виконати вимірювання стану механічного або електронного контактного перетворювача. Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Визначення зміни стану контактного перетворювача за допомогою модуля Trigger

1. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Time of filtration** обрати період часу між двома послідовними вимірюваннями (задається в сотнях мікросекунд – наприклад, значення 5 параметра **Time of filtration** відповідає періоду часу між двома послідовними вимірюваннями 500 мкс ).

3. В лівій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** додати модуль **Trigger [Var]**.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Trigger [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Number of input** обрати номер дискретного входу **DI–2–0**.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Trigger [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Sense edge** обрати фронт вхідного дискретного сигналу, по якому буде спрацьовувати модуль **Trigger** (наростаючий **RISE_EDGE**, спадний **FALL_EDGE** або одночасно наростаючий і спадний **BOTH_EDGE**).

6. Виконати вимірювання вихідного сигналу модуля **Trigger [Var]**, змінюючи стан механічного або електронного контактного перетворювача. Ре-

зультат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

7. Зіставити стану механічного або електронного контактного перетворювача значення певного технологічного параметра (наприклад, положення заслінки “Зачинено/Відчинено”, положення регулюючого клапана “Крайнє ліве/ Крайнє праве”, стан засклення “Розбите/Не розбите”, стан периметра охорони “Порушений/Не порушений” тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3. Визначення Counter 16bit**

1. Підключити до дискретного входу **DI–3–0** механічний або електронний контактний перетворювач.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Time of filtration** обрати період часу між двома послідовними вимірюваннями (задається в сотнях мікросекунд – наприклад, значення 5 параметра **Time of filtration** відповідає періоду часу між двома послідовними вимірюваннями 500 мкс ).

3. В лівій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** додати модуль **Counter 16bit [Var]**.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Counter 16bit [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Number of input** обрати номер дискретного входу **DI–3–0**.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Counter 16bit [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Sense edge** обрати фронт вхідного дискретного сигналу, по якому буде спрацьовувати модуль **Counter 16bit [Var]** (наростаючий **RISE_EDGE**, спадний **FALL_EDGE** або одночасно наростаючий і спадний **BOTH_EDGE**).

6. Виконати вимірювання вихідного сигналу модуля **Counter 16bit [Var]**, змінюючи стан механічного або електронного контактного перетворювача. Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

7. Зіставити стану механічного або електронного контактного перетворювача значення певного технологічного параметра (наприклад, кількість обертів електричного привода, кількість одиниць продукції на конвеєрі, кількість відкриттів/закриттів воріт, кількість проходжень персоналу через турнікет, кількість спрацьовувань звукової або світлової сигналізації тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.4. Визначення Encoder 16bit**

1. Підключити до дискретних входів **DI–4–0** і **DI–5–0** механічні або електронні контактні перетворювачі.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** на вкладці **Module parameters** в полі **Value** для пара-

метра **Time of filtration** обрати період часу між двома послідовними вимірюваннями (задається в сотнях мікросекунд – наприклад, значення 5 параметра **Time of filtration** відповідає періоду часу між двома послідовними вимірюваннями 500 мкс ).

3. В лівій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** додати модуль **Encoder 16bit [Var]**.

4. В правій частині вікна редактора конфігуратора ПЛК для модуля **Encoder 16bit [Var]** на вкладці **Module parameters** в полі **Value** для параметра **First input** обрати номер дискретного входу **DI–4–0**, а в полі **Value** для параметра **Second input** обрати номер дискретного входу **DI–5–0**,

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Encoder 16bit [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Range** обрати кількість імпульсів дискретного сигналу, яка відповідає одному повному обороту кругового енкодера або одному повному ходу лінійного енкодера.

6. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Encoder 16bit [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Encoder type** обрати тип енкодера (круговий **RING** або лінійний **LINEAR**).

7. Виконати вимірювання вихідного сигналу модуля **Encoder 16bit [Var]**, змінюючи стан механічних або електронних контактних перетворювачів. Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

8. Зіставити стану механічного або електронного контактного перетворювача значення певного технологічного параметра (наприклад, кутове положення заслінки, лінійне положення регулюючого клапана, лінійне положення кареток станка з числовим програмним керуванням, кутове положення антени радіолокатора, кутове положення поворотного стола тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.5. Визначення Counter SP

1. Підключити до дискретних входів **DI–4–0**, **DI–5–0** і **DI–6–0** механічні або електронні контактні перетворювачі.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Time of filtration** обрати період часу між двома послідовними вимірюваннями (задається в сотнях мікросекунд – наприклад, значення 5 параметра **Time of filtration** відповідає періоду часу між двома послідовними вимірюваннями 500 мкс ).

3. В лівій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete input 6 bit [Fix]** додати модуль **Counter SP [Var]**.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Counter SP [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Start pin num** обрати номер дискретного входу **DI–4–0**, в полі **Value** для параметра **Stop pin num** обрати номер дискретного входу **DI–5–0**, а полі **Value** для

параметра **Count pin num** обрати номер дискретного входу **DI–6–0**.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Counter SP [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Start sense edge** обрати фронт вхідного дискретного сигналу, по якому буде починатися підрахунок імпульсів дискретного сигналу модулем **Counter SP [Var]** (наростаючий **Rising** або спадний **Falling**), в полі **Value** для параметра **Stop sense edge** обрати фронт вхідного дискретного сигналу, по якому буде зупинятися підрахунок імпульсів дискретного сигналу модулем **Counter SP [Var]** (наростаючий **Rising** або спадний **Falling**), а в полі **Value** для параметра **Count sense edge** обрати фронт вхідного дискретного сигналу, по якому буде спрацьовувати модуль **Counter SP [Var]** (наростаючий **Rising** або спадний **Falling**).

6. Виконати вимірювання вихідного сигналу модуля **Counter SP [Var]**, змінюючи стан механічних або електронних контактних перетворювачів. Результат вимірювання відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

7. Зіставити стану механічного або електронного контактного перетворювача значення певного технологічного параметра (наприклад, кількість автомобільного транспорту (який заїжджав на територію підприємства і виїжджав з неї протягом певного проміжку часу), кількість вмикань/вимикань технологічного обладнання протягом певного проміжку часу тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Як можна підключити до ПЛК150 механічні контактні перетворювачі (“сухі” контакти)?
2. Як можна підключити до ПЛК150 електронні контактні перетворювачі ( n – p – n -транзистори з відкритим колектором)?
3. Яка напруга на дискретному вході ПЛК150 відповідає лог. 1?
4. Яка напруга на дискретному вході ПЛК150 відповідає лог. 0?
5. Якою повинна бути максимальна частота імпульсів дискретного сигналу, який подається на дискретний вхід ПЛК150?
6. Як працює модуль **Trigger [Var]**?
7. Як працює модуль **Counter 16bit [Var]**?
8. Як працює модуль **Encoder 16bit [Var]**?
9. Як працює модуль **Counter SP [Var]**?
10. Як у вікні редактора *Конфігуратора ПЛК* можна обрати період часу між двома послідовними вимірюваннями?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №15.
# ВИВЧЕННЯ РОБОТИ АНАЛОГОВИХ ВИХОДІВ ПЛК150 ВИРОБНИЦТВА ТОВ “ВО ОВЕН”

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи аналогових виходів ПЛК150 виробництва ТОВ “ВО ОВЕН”.

## 2. Порядок виконання лабораторної роботи

### 2.1. Формування уніфікованого електричного сигналу в вигляді напруги

1. Підключити до аналогового виходу **АО–1–0** активне (резистивне) навантаження.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Name** для параметра **Type o**обрати значення **Voltage 0-10 В**, яке відповідає вихідній уніфікованій електричній напрузі $(0\ldots10)$ В.

3. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Null correction** (за необхідності) обрати значення напруги, на яке буде виконане коригування результату формування (параметр **Null correction** відповідає напрузі 1 В).

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Full range correction** (за необхідності) обрати значення напруги, на яке буде виконане коригування результату формування (параметр **Full range correction** відповідає напрузі 10 В).

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Safe value** (за необхідності) обрати значення сформованої напруги “за замовчуванням”.

6. Виконати вимірювання уніфікованої електричної напруги (наприклад, за допомогою аналогового вольтметра, цифрового вольтметра, мультиметра тощо). Сформовану уніфіковану електричну напругу відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

7. Зіставити уніфікованій електричній напрузі значення певного технологічного параметра (наприклад, кута повороту засувки, обертів електричного двигуна тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Формування уніфікованого електричного сигналу в вигляді струму

1. Підключити до аналогового виходу **АО–2–0** активне (резистивне) на-

вантаження.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Name** для параметра **Type o**обрати значення **Current 4-20 мА**, яке відповідає вихідному уніфікованому електричному струму $(4\ldots20)$ мА.

3. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Null correction** (за необхідності) обрати значення струму, на яке буде виконане коригування результату формування (параметр **Null correction** відповідає струму 4 мА).

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Full range correction** (за необхідності) обрати значення струму, на яке буде виконане коригування результату формування (параметр **Full range correction** відповідає струму 15 мА).

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Analog output [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Safe value** (за необхідності) обрати значення сформованого струму "за замовчуванням".

6. Виконати вимірювання уніфікованого електричного струму (наприклад, за допомогою аналогового амперметра, цифрового амперметра, мультиметра тощо). Сформований уніфікований електричний струм відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

7. Зіставити уніфікованому електричному струму значення певного технологічного параметра (наприклад, кута повороту засувки, обертів електричного двигуна тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Які існують уніфіковані електричні напруги (згідно з [2])?
2. Які існують уніфіковані електричні струми (згідно з [2])?
3. Як можна підключити до ПЛК150 активне (резистивне) навантаження?
4. Як можна сформувати на виході ПЛК150 уніфіковану електричну напругу?
5. Як можна сформувати на виході ПЛК150 уніфікований електричний струм?
6. Як і на що впливають параметри **Null correction** і **Full range correction**?
7. Як можна обрати значення сформованої напруги або сформованого струму "за замовчуванням"?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №16.
# ВИВЧЕННЯ РОБОТИ ДИСКРЕТНИХ ВИХОДІВ ПЛК150 ВИРОБНИЦТВА ТОВ “ВО ОВЕН”

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи дискретних виходів ПЛК150 виробництва ТОВ “ВО ОВЕН”.

## 2. Порядок виконання лабораторної роботи

### 2.1. Формування стану електромагнітного реле

1. Підключити до дискретного виходу **DO–1–0** активне (резистивне) або реактивне (ємнісне, індуктивне, ємнісно-індуктивне тощо) навантаження.

2. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete output 4 bit [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Safe value** (за необхідності) обрати стан електромагнітного реле “за замовчуванням”.

3. Виконати вимірювання стану електромагнітного реле (наприклад, за допомогою аналогового вольтметра, цифрового вольтметра, аналогового амперметра, цифрового амперметра, мультиметра тощо). Стан електромагнітного реле відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

4. Зіставити стану електромагнітного реле значення певного технологічного параметра (наприклад, кінцевого положення засувки, напряму обертів електричного двигуна, підключення до електричної мережі електричного нагрівача тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Формування роботи електромагнітного реле в режимі широтно-імпульсної модуляції

1. Підключити до дискретного виходу **DO–2–0** активне (резистивне) або реактивне (ємнісне, індуктивне, ємнісно-індуктивне тощо) навантаження.

2. В лівій частині вікна редактора *Конфігуратора ПЛК* для модуля **Discrete output 4 bit [Fix]** додати модуль **Pulse-wide modulator [Var]**.

3. В правій частині вікна редактора конфігуратора ПЛК для модуля **Discrete output 4 bit [Fix]** на вкладці **Module parameters** в полі **Value** для параметра **Safe value** (за необхідності) обрати стан електромагнітного реле “за замовчуванням”.

4. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Pulse-wide modulator [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Number of output** обрати номер дискретного виходу **DO–2–0**.

5. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Pulse-wide modulator [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Period of PWM in 100 mksec** обрати період часу між двома послідовними імпульсами (задається в сотнях мікросекунд – наприклад, значення 5 па-

раметра **Period of PWM in 100 mksec** відповідає періоду часу між двома послідовними імпульсами $500$ мкс).

6. В правій частині вікна редактора *Конфігуратора ПЛК* для модуля **Pulse-wide modulator [Var]** на вкладці **Module parameters** в полі **Value** для параметра **Minimal duration of impulse in 100 mksec** обрати мінімально допустиму тривалість імпульсу (задається в сотнях мікросекунд – наприклад, значення $2$ параметра **Minimal duration of impulse in 100 mksec** відповідає мінімально допустимій тривалості імпульсу $200$ мкс).

7. Виконати вимірювання стану електромагнітного реле (наприклад, за допомогою аналогового вольтметра, цифрового вольтметра, аналогового амперметра, цифрового амперметра, мультиметра тощо). Стан електромагнітного реле відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

8. Зіставити стану електромагнітного реле значення певного технологічного параметра (наприклад, кінцевого положення засувки, напряму обертів електричного двигуна, підключення до електричної мережі електричного нагрівача тощо). Відобразити значення цього технологічного параметра як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Яке значення має поняття “нормально замкнений контакт”?
2. Яке значення має поняття “нормально розімкнений контакт”?
3. Яке максимальне значення повинна мати частота перемикання електромагнітного реле для його нормального функціонування протягом середнього строку служби ПЛК?
4. Яку максимальну постійну або змінну електричну напругу може витримувати електромагнітне реле?
5. Який максимальний постійний або змінний електричний струм може витримувати електромагнітне реле?
6. Яке допускається мінімальне значення $\cos(\varphi)$ при комутації електромагнітним реле змінних сигналів?
7. Як працює модуль **Pulse-wide modulator [Var]**?
8. Як у вікні редактора *Конфігуратора ПЛК* обрати період часу між двома послідовними імпульсами?
9. Як у вікні редактора *Конфігуратора ПЛК* мінімально допустиму тривалість імпульсу?
10. Як у вікні редактора *Конфігуратора ПЛК* можна обрати стан електромагнітного реле “за замовчуванням”?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №17.
# ВИВЧЕННЯ РОБОТИ БІСТАБІЛЬНИХ ФУНКЦІОНАЛЬНИХ БЛОКІВ RS, SEMA І SR

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи бістабільних функціональних блоків **RS**, **SEMA** і **SR**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Формування вихідного сигналу за допомогою бістабільного функціонального блока RS

#### 2.1.1. Формування вихідного сигналу за допомогою бістабільного функціонального блока RS з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.
3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
4. Створити екземпляр **_RS_** функціонального блока **RS**.
5. Створити змінну **_SET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
6. Створити змінну **_RESET1_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).
7. Створити змінну **_Q1_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).
8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _SET_
ST      _RS_.SET
LD      _RESET1_
ST      _RS_.RESET1
CAL     _RS_
LD      _RS_.Q1
ST      _Q1_
```

Запустити цей програмний код на виконання.

9. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.2. Формування вихідного сигналу за допомогою бістабільного функціонального блока RS з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

4. Створити екземпляр **_RS_** функціонального блока **RS**.

5. Створити змінну **_SET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET1_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q1_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_RS_(SET:=_SET_, RESET1:=_RESET1_) ;
_Q1_ := _RS_.Q1 ;
```

Запустити цей програмний код на виконання.

9. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.3. Формування вихідного сигналу за допомогою бістабільного функціонального блока RS з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

4. Створити екземпляр **_RS_** функціонального блока **RS**.

5. Створити змінну **_SET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET1_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q1_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Програмно зв’язати змінну **_SET_** із входом **SET:BOOL** екземпляра **_RS_** функціонального блока **RS** (див. рис. 17.1).

9. Програмно зв’язати змінну **_RESET1_** із входом **RESET1:BOOL** екземпляра **_RS_** функціонального блока **RS** (див. рис. 17.1).

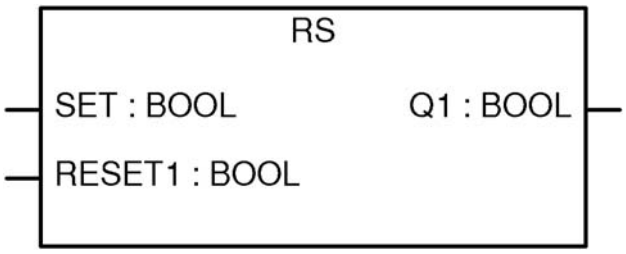



Рис. 17.1. Бістабільний функціональний блок **RS**

10. Програмно зв'язати змінну **_Q1_** із виходом **Q1:BOOL** екземпляра **_RS_** функціонального блока **RS** (див. рис. 17.1).

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Формування вихідного сигналу за допомогою бістабільного функціонального блока SEMA

#### 2.2.1. Формування вихідного сигналу за допомогою бістабільного функціонального блока SEMA з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

4. Створити екземпляр **_SEMA_** функціонального блока **SEMA**.

5. Створити змінну **_CLAIM_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RELEASE_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_BUSY_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _CLAIM_
ST      _SEMA_.CLAIM
LD      _RELEASE_
ST      _SEMA_.RELEASE
CAL     _SEMA_
LD      _SEMA_.BUSY
ST      _BUSY_
```

Запустити цей програмний код на виконання.

9. Змінюючи стан механічних або електронних контактних перетворюва-

чів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.2. Формування вихідного сигналу за допомогою бістабільного функціонального блока SEMA з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

4. Створити екземпляр **_SEMA_** функціонального блока **SEMA**.

5. Створити змінну **_CLAIM_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RELEASE_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_BUSY_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_SEMA_(CLAIM:=_CLAIM_, RELEASE:=_RELEASE_) ;
_BUSY_ := _SEMA_.BUSY ;
```

Запустити цей програмний код на виконання.

9. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.3. Формування вихідного сигналу за допомогою бістабільного функціонального блока SEMA з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

4. Створити екземпляр **_SEMA_** функціонального блока **SEMA**.

5. Створити змінну **_CLAIM_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RELEASE_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_BUSY_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Програмно зв’язати змінну **_CLAIM_** із входом **CLAIM:BOOL** екземпляра **_SEMA_** функціонального блока **SEMA** (див. рис. 17.2).

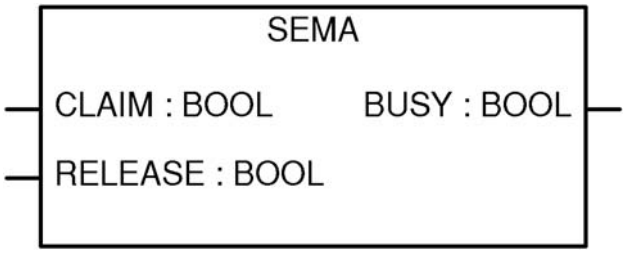



Рис. 17.2. Бістабільний функціональний блок **SEMA**

9. Програмно зв’язати змінну **_RELEASE_** із входом **RELEASE:BOOL** екземпляра **_SEMA_** функціонального блока **SEMA** (див. рис. 17.2).

10. Програмно зв’язати змінну **_ BUSY _** із виходом **BUSY:BOOL** екземпляра **_SEMA_** функціонального блока **SEMA** (див. рис. 17.2).

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.3. Формування вихідного сигналу за допомогою бістабільного функціонального блока SR

#### 2.3.1. Формування вихідного сигналу за допомогою бістабільного функціонального блока SR з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

4. Створити екземпляр **_SR_** функціонального блока **SR**.

5. Створити змінну **_SET1_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q1_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _SET1_
ST      _SR_.SET1
LD      _RESET_
ST      _SR_.RESET
```

```
CAL     _SR_
LD      _SR_.Q1
ST      _Q1_
```

Запустити цей програмний код на виконання.

9. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.2. Формування вихідного сигналу за допомогою бістабільного функціонального блока SR з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

4. Створити екземпляр **_SR_** функціонального блока **SR**.

5. Створити змінну **_SET1_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q1_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_SR_(SET1:=_SET1_, RESET:=_RESET_) ;
_Q1_ := _SR_.Q1 ;
```

Запустити цей програмний код на виконання.

9. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.3. Формування вихідного сигналу за допомогою бістабільного функціонального блока SR з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

4. Створити екземпляр **_SR_** функціонального блока **SR**.

5. Створити змінну **_SET1_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q1_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Програмно зв'язати змінну **_SET1_** із входом **SET1:BOOL** екземпляра **_SR_** функціонального блока **SR** (див. рис. 17.3).

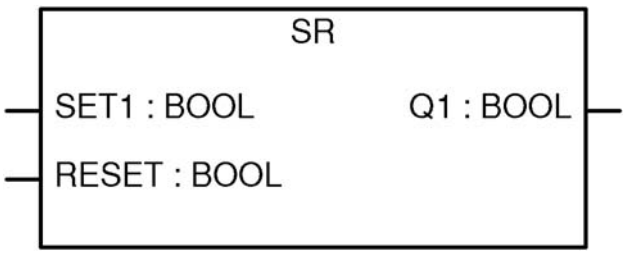



Рис. 17.3. Бістабільний функціональний блок **SR**

9. Програмно зв'язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_SR_** функціонального блока **SR** (див. рис. 17.3).

10. Програмно зв'язати змінну **_Q1_** із виходом **Q1:BOOL** екземпляра **_SR_** функціонального блока **SR** (див. рис. 17.3).

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Які дискретні входи має ПЛК150?
2. Як подати на дискретний вхід ПЛК150 лог. 1?
3. Як подати на дискретний вхід ПЛК150 лог. 0?
4. Які дискретні виходи має ПЛК150?
5. Як створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG**?
6. Як створити екземпляр функціонального блока?
7. Як створити змінну із заданою адресою?
8. Як працює бістабільний функціональний блок **RS**?
9. Як працює бістабільний функціональний блок **SEMA**?
10. Як працює бістабільний функціональний блок **SR**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №18.
# ВИВЧЕННЯ РОБОТИ ЛІЧИЛЬНИКІВ CTD, CTU І CTUD

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи лічильників **CTD**, **CTU** і **CTUD**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTD

#### 2.1.1. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTD з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.
3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
4. Створити екземпляр **_CTD_** функціонального блока **CTD**.
5. Створити змінну **_CD_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
6. Створити змінну **_LOAD_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).
7. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).
8. Створити змінну **_CV_** типу **WORD**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _CD_
ST      _CTD_.CD
LD      _LOAD_
ST      _CTD_.LOAD
LD      60
ST      _CTD_.PV
CAL     _CTD_
LD      _CTD_.Q
ST      _Q_
LD      _CTD_.CV
ST      _CV_
```

10. Запустити цей програмний код на виконання.
11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретного

виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.2. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTD з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.
3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.
4. Створити екземпляр **_CTD_** функціонального блока **CTD**.
5. Створити змінну **_CD_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
6. Створити змінну **_LOAD_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).
7. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).
8. Створити змінну **_CV_** типу **WORD**.
9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_CTD_(CD:=_CD_, LOAD:=_LOAD_, PV:=60) ;
_Q_ := _CTD_.Q ;
_CV_ := _CTD_.CV ;
```

10. Запустити цей програмний код на виконання.
11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.3. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTD з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.
3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.
4. Створити екземпляр **_CTD_** функціонального блока **CTD**.
5. Створити змінну **_CD_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_LOAD_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_CV_** типу **WORD**.

9. Програмно зв'язати змінну **_CD_** із входом **CD:BOOL** екземпляра **_CTD_** функціонального блока **CTD** (див. рис. 18.1).

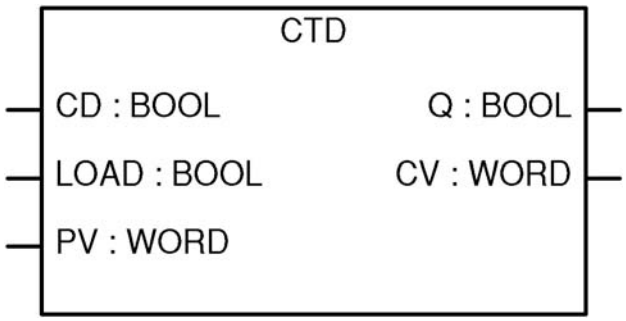



Рис. 18.1. Функціональний блок лічильник **CTD**

10. Програмно зв'язати змінну **_LOAD_** із входом **LOAD:BOOL** екземпляра **_CTD_** функціонального блока **CTD** (див. рис. 18.1).

11. Подати на вхід **PV:WORD** значення **60**.

12. Програмно зв'язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_CTD_** функціонального блока **CTD** (див. рис. 18.1).

13. Програмно зв'язати змінну **_CV_** із виходом **CV:WORD** екземпляра **_CTD_** функціонального блока **CTD** (див. рис. 18.1).

14. Запустити цей програмний код на виконання.

15. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTU

#### 2.2.1. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTU з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

4. Створити екземпляр **_CTU_** функціонального блока **CTU**.

5. Створити змінну **_CU_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_CV_** типу **WORD**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _CU_
ST      _CTU_.CU
LD      _RESET_
ST      _CTU_.RESET
LD      60
ST      _CTU_.PV
CAL     _CTU_
LD      _CTU_.Q
ST      _Q_
LD      _CTU_.CV
ST      _CV_
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2.2. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTU з використанням мови програмування Structured Text (ST)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

4. Створити екземпляр **_CTU_** функціонального блока **CTU**.

5. Створити змінну **_CU_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_CV_** типу **WORD**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_CTU_(CU:=_CU_, RESET:=_RESET_, PV:=60) ;
_Q_ := _CTU_.Q ;
_CV_ := _CTU_.CV;
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2.3. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTU з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

4. Створити екземпляр **_CTD_** функціонального блока **CTD**.

5. Створити змінну **_CU_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_CV_** типу **WORD**.

9. Програмно зв’язати змінну **_CU_** із входом **CU:BOOL** екземпляра **_CTU_** функціонального блока **CTU** (див. рис. 18.2).

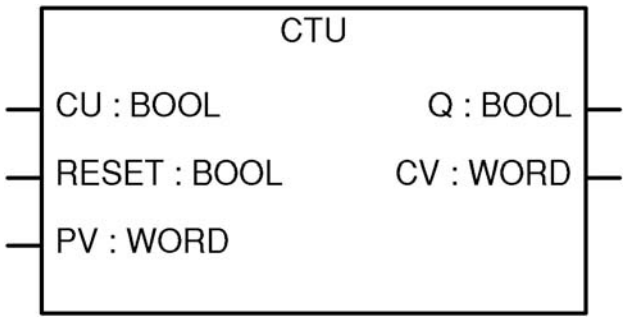



Рис. 18.2. Лічильник **CTU**

10. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_CTU_** функціонального блока **CTU** (див. рис. 18.2).

11. Подати на вхід **PV:WORD** значення **60**.

12. Програмно зв’язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_CTU_** функціонального блока **CTU** (див. рис. 18.2).

13. Програмно зв’язати змінну **_CV_** із виходом **CV:WORD** екземпляра **_CTU_** функціонального блока **CTU** (див. рис. 18.2).

14. Запустити цей програмний код на виконання.

15. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.3. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTUD

#### 2.3.1. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTUD з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Підключити до дискретного входу **DI–3–0** механічний або електронний контактний перетворювач.

4. Підключити до дискретного входу **DI–4–0** механічний або електронний контактний перетворювач.

5. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

6. Створити екземпляр **_CTUD_** функціонального блока **CTUD**.

7. Створити змінну **_CU_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

8. Створити змінну **_CD_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

9. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.2** (яка відповідає дискретному входу **DI–3–0**).

10. Створити змінну **_LOAD_** типу **BOOL** із адресою **%IX0.3** (яка відповідає дискретному входу **DI–4–0**).

11. Створити змінну **_QU_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

12. Створити змінну **_QD_** типу **BOOL** із адресою **%QX2.0** (яка відповідає дискретному виходу **DO–2–0**).

13. Створити змінну **_CV_** типу **WORD**.

14. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _CU_
ST      _CTUD_.CU
LD      _CD_
ST      _CTUD_.CD
```

```
LD          _RESET_
ST          _CTUD_.RESET
LD          _LOAD_
ST          _CTUD_.LOAD
LD          60
ST          _CTUD_.PV
CAL         _CTUD_
LD          _CTUD_.QU
ST          _QU_
LD          _CTUD_.QD
ST          _QD_
LD          _CTUD_.CV
ST          _CV_
```

15. Запустити цей програмний код на виконання.

16. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0**, **DI–2–0**, **DI–3–0** і **DI–4–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле), а також значення змінної **_CV_**. Стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.2. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTUD з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Підключити до дискретного входу **DI–3–0** механічний або електронний контактний перетворювач.

4. Підключити до дискретного входу **DI–4–0** механічний або електронний контактний перетворювач.

5. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

6. Створити екземпляр **_CTUD_** функціонального блока **CTUD**.

7. Створити змінну **_CU_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

8. Створити змінну **_CD_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

9. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.2** (яка відповідає дискретному входу **DI–3–0**).

10. Створити змінну **_LOAD_** типу **BOOL** із адресою **%IX0.3** (яка відповідає дискретному входу **DI–4–0**).

11. Створити змінну **_QU_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

12. Створити змінну **_QD_** типу **BOOL** із адресою **%QX2.0** (яка відповідає дискретному виходу **DO–2–0**).

13. Створити змінну **_CV_** типу **WORD**.

14. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_CTUD_(CU:=_CU_,        CD:=_CD_,        RESET:=_RESET_,
LOAD:=_LOAD_, PV:=60) ;
_QU_ := _CTUD_.QU ;
_QD_ := _CTUD_.QD ;
_CV_ := _CTUD_.CV ;
```

15. Запустити цей програмний код на виконання.

16. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0**, **DI–2–0**, **DI–3–0** і **DI–4–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле), а також значення змінної **_CV_**. Стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.3.3. Підрахунок кількості імпульсів за допомогою функціонального блока лічильник CTUD з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Підключити до дискретного входу **DI–3–0** механічний або електронний контактний перетворювач.

4. Підключити до дискретного входу **DI–4–0** механічний або електронний контактний перетворювач.

5. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

6. Створити екземпляр **_CTUD_** функціонального блока **CTUD**.

7. Створити змінну **_CU_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

8. Створити змінну **_CD_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

9. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.2** (яка відповідає дискретному входу **DI–3–0**).

10. Створити змінну **_LOAD_** типу **BOOL** із адресою **%IX0.3** (яка відповідає дискретному входу **DI–4–0**).

11. Створити змінну **_QU_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

12. Створити змінну **_QD_** типу **BOOL** із адресою **%QX2.0** (яка відповідає дискретному виходу **DO–2–0**).

13. Створити змінну **_CV_** типу **WORD**.

14. Програмно зв’язати змінну **_CU_** із входом **CU:BOOL** екземпляра

**_CTU_** функціонального блока **CTU** (див. рис. 18.3).

15. Програмно зв’язати змінну **_CD_** із входом **CD:BOOL** екземпляра **_CTUD_** функціонального блока **CTUD** (див. рис. 18.3).

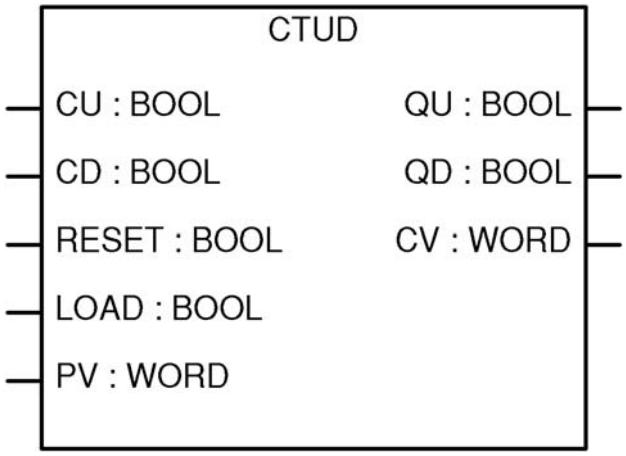



Рис. 18.3. Лічильник **CTUD**

16. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_CTUD_** функціонального блока **CTUD** (див. рис. 18.3).

17. Програмно зв’язати змінну **_LOAD_** із входом **LOAD:BOOL** екземпляра **_CTUD_** функціонального блока **CTUD** (див. рис. 18.3).

18. Подати на вхід **PV:WORD** значення **60**.

19. Програмно зв’язати змінну **_QU_** із виходом **QU:BOOL** екземпляра **_CTUD_** функціонального блока **CTUD** (див. рис. 18.3).

20. Програмно зв’язати змінну **_QD_** із виходом **QD:BOOL** екземпляра **_CTUD_** функціонального блока **CTUD** (див. рис. 18.3).

21. Програмно зв’язати змінну **_CV_** із виходом **CV:WORD** екземпляра **_CTU_** функціонального блока **CTU** (див. рис. 18.3).

22. Запустити цей програмний код на виконання.

23. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0**, **DI–2–0**, **DI–3–0** і **DI–4–0**, а також значення, яке подається на вхід **PV**, виконати вимірювання стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле), а також значення змінної **_CV_**. Стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле), а також значення змінної **_CV_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Як працює лічильник **CTD**?
2. Як працює лічильник **CTU**?
3. Як працює лічильник **CTUD**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №19.
# ВИВЧЕННЯ РОБОТИ ТАЙМЕРІВ RTC, TOF, TON І TP

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи таймерів **RTC**, **TOF**, **TON** і **TP**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Вимірювання і формування періодів часу за допомогою функціонального блока таймер RTC

#### 2.1.1. Вимірювання і формування періодів часу за допомогою функціонального блока таймер RTC з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
3. Створити екземпляр **_RTC_** функціонального блока **RTC**.
4. Створити змінну **_EN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).
6. Створити змінну **_CDT_** типу **DT**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _EN_
ST      _RTC_.EN
LD      DT#2010-01-01-10:00:00
ST      _RTC_.PDT
CAL     _RTC_
LD      _RTC_.Q
ST      _Q_
LD      _RTC_.CDT
ST      _CDT_
```

8. Запустити цей програмний код на виконання.
9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PDT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CDT_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CDT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.2. Вимірювання і формування періодів часу за допомогою функціонального блока таймер RTC з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_RTC_** функціонального блока **RTC**.

4. Створити змінну **_EN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_CDT_** типу **DT**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_RTC_(EN:=_EN_, PDT:=DT#2010-01-01-10:00:00) ;
_Q_ := _RTC_.Q ;
_CDT_ := _RTC_.CDT ;
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PDT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CDT_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CDT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.3. Вимірювання і формування періодів часу за допомогою функціонального блока таймер RTC з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_RTC_** функціонального блока **RTC**.

4. Створити змінну **_EN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_CDT_** типу **DT**.

7. Програмно зв’язати змінну **_EN_** із входом **EN:BOOL** екземпляра **_RTC_** функціонального блока **RTC** (див. рис. 19.1).

8. Подати на вхід **PDT:DT** значення **DT#2010-01-01-10:00:00**.

9. Програмно зв’язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_RTC_** функціонального блока **RTC** (див. рис. 19.1).

10. Програмно зв’язати змінну **_CDT_** із виходом **CDT:DT** екземпляра **_RTC_** функціонального блока **RTC** (див. рис. 19.1).

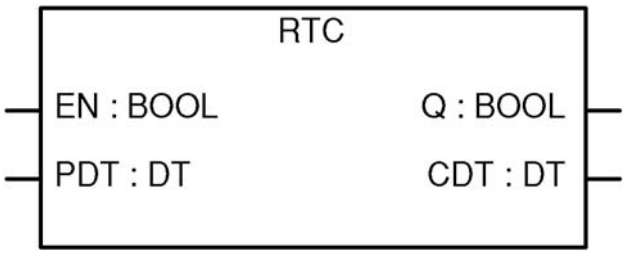



Рис. 19.1. Функціональний блок таймер **RTC**

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PDT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CDT_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_CDT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TOF

#### 2.2.1. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TOF з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_TOF_** функціонального блока **TOF**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_
ST      _TOF_.IN
LD      T#5s
ST      _TOF_.PT
CAL     _TOF_
LD      _TOF_.Q
ST      _Q_
LD      _TOF_.ET
ST      _ET_
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке пода-

ється на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.2. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TOF з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_TOF_** функціонального блока **TOF**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_TOF_(IN:=_IN_, PT:=T#5s) ;
_Q_ := _TOF_.Q ;
_ET_ := _TOF_.ET ;
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.3. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TOF з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_TOF_** функціонального блока **TOF**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Програмно зв’язати змінну **_IN_** із входом **IN:BOOL** екземпляра **_TOF_** функціонального блока **TOF** (див. рис. 19.2).

8. Подати на вхід **PT:TIME** значення **T#5s**.

9. Програмно зв’язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_TOF_** функціонального блока **TOF** (див. рис. 19.2).

10. Програмно зв’язати змінну **_ET_** із виходом E**T:TIME** екземпляра **_TOF_** функціонального блока **TOF** (див. рис. 19.2).

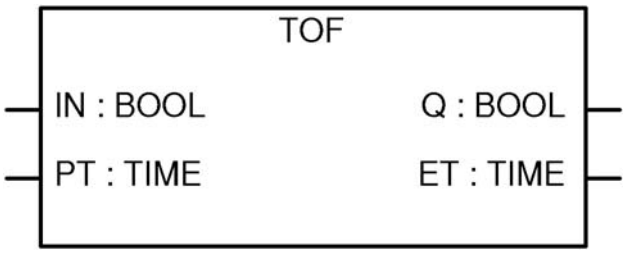



Рис. 19.2. Функціональний блок таймер **TOF**

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.3. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TON

#### 2.3.1. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TON з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_TON_** функціонального блока **TON**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_
ST      _TON_.IN
LD      T#5s
ST      _TON_.PT
CAL     _TON_
LD      _TON_.Q
ST      _Q_
```

**LD** **_TON_.ET**
**ST** **_ET_**

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.2. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TON з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_TON_** функціонального блока **TON**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_TON_(IN:=_IN_, PT:=T#5s) ;
_Q_ := _TON_.Q ;
_ET_ := _TON_.ET ;
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.3. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TON з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_TON_** функціонального блока **TON**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Програмно зв'язати змінну **_IN_** із входом **IN:BOOL** екземпляра **_TON_** функціонального блока **TON** (див. рис. 19.3).

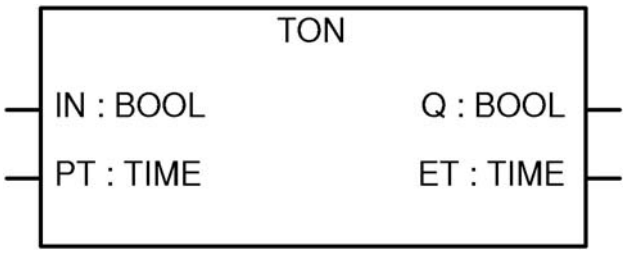



Рис. 19.3. Функціональний блок таймер **TON**

8. Подати на вхід **PT:TIME** значення **T#5s**.

9. Програмно зв'язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_TON_** функціонального блока **TON** (див. рис. 19.3).

10. Програмно зв'язати змінну **_ET_** із виходом **ET:TIME** екземпляра **_TON_** функціонального блока **TON** (див. рис. 19.3).

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.4. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TP

#### 2.4.1. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TP з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_TP_** функціонального блока **TP**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_
ST      _TP_.IN
```

```
LD      T#5s
ST      _TP_.PT
CAL     _TP_
LD      _TP_.Q
ST      _Q_
LD      _TP_.ET
ST      _ET_
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.4.2. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TP з використанням мови програмування Structured Text (ST)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_TP_** функціонального блока **TP**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_TP_(IN:=_IN_, PT:=T#5s) ;
_Q_ := _TP_.Q ;
_ET_ := _TP_.ET ;
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.4.3. Вимірювання і формування періодів часу за допомогою функціонального блока таймер TP з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний

контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_TP_** функціонального блока **TP**.

4. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Створити змінну **_ET_** типу **TIME**.

7. Програмно зв'язати змінну **_IN_** із входом **IN:BOOL** екземпляра **_TP_** функціонального блока **TP** (див. рис. 19.4).

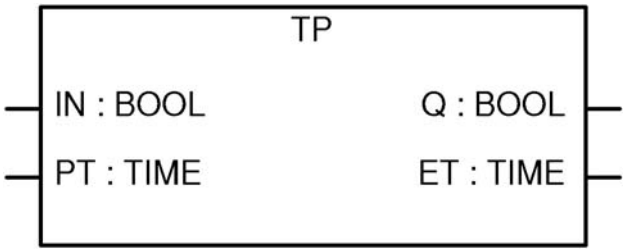



Рис. 19.4. Функціональний блок таймер **TP**

8. Подати на вхід **PT:TIME** значення **T#5s**.

9. Програмно зв'язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_TP_** функціонального блока **TP** (див. рис. 19.4).

10. Програмно зв'язати змінну **_ET_** із виходом **ET:TIME** екземпляра **_TP_** функціонального блока **TP** (див. рис. 19.4).

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, яке подається на вхід **PT**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_**. Стан дискретного виходу **DO–1–0** (електромагнітного реле), а також значення змінної **_ET_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Як працює таймер **RTC**?
2. Як працює таймер **TOF**?
3. Як працює таймер **TON**?
4. Як працює таймер **TP**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №20.
# ВИВЧЕННЯ РОБОТИ ТРИГЕРІВ F_TRIG І R_TRIG

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи тригерів **F_TRIG** і **R_TRIG**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Формування вихідного сигналу за допомогою функціонального блока тригер F_TRIG

#### 2.1.1. Формування вихідного сигналу за допомогою функціонального блока тригер F_TRIG з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
3. Створити екземпляр **_F_TRIG_** функціонального блока **F_TRIG**.
4. Створити змінну **_CLK_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).
6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _CLK_
ST      _F_TRIG_.CLK
CAL     _F_TRIG_
LD      _F_TRIG_.Q
ST      _Q_
```

7. Запустити цей програмний код на виконання.
8. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

#### 2.1.2. Формування вихідного сигналу за допомогою функціонального блока тригер F_TRIG з використанням мови програмування Structured Text (ST)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.
3. Створити екземпляр **_F_TRIG_** функціонального блока **F_TRIG**.

4. Створити змінну **_CLK_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_F_TRIG_(CLK:=_CLK_) ;
_Q_ := _F_TRIG_.Q ;
```

7. Запустити цей програмний код на виконання.

8. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.1.3. Формування вихідного сигналу за допомогою функціонального блока тригер F_TRIG з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_F_TRIG_** функціонального блока **F_TRIG**.

4. Створити змінну **_CLK_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Програмно зв’язати змінну **_CLK_** із входом **CLK:BOOL** екземпляра **_F_TRIG_** функціонального блока **F_TRIG** (див. рис. 20.1).


Рис. 20.1. Функціональний блок тригер **F_TRIG**

7. Програмно зв’язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_F_TRIG_** функціонального блока **F_TRIG** (див. рис. 20.1).

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2. Формування вихідного сигналу за допомогою функціонального блока тригер R_TRIG**

**2.2.1. Формування вихідного сигналу за допомогою функціонального блока тригер R_TRIG з використанням мови програмування Instruction List (IL)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_R_TRIG_** функціонального блока **R_TRIG**.

4. Створити змінну **_CLK_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _CLK_
ST      _R_TRIG_.CLK
CAL     _R_TRIG_
LD      _R_TRIG_.Q
ST      _Q_
```

7. Запустити цей програмний код на виконання.

8. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.2. Формування вихідного сигналу за допомогою функціонального блока тригер R_TRIG з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_R_TRIG_** функціонального блока **R_TRIG**.

4. Створити змінну **_CLK_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_R_TRIG_(CLK:=_CLK_) ;
_Q_ := _R_TRIG_.Q ;
```

7. Запустити цей програмний код на виконання.

8. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.3. Формування вихідного сигналу за допомогою функціонального блока тригер R_TRIG з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_R_TRIG_** функціонального блока **R_TRIG**.

4. Створити змінну **_CLK_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_Q_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Програмно зв’язати змінну **_CLK_** із входом **CLK:BOOL** екземпляра **_R_TRIG_** функціонального блока **R_TRIG** (див. рис. 20.2).

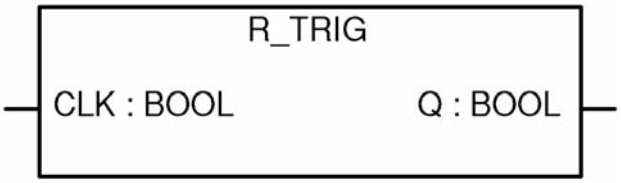



Рис. 20.2. Функціональний блок тригер **R_TRIG**

7. Програмно зв’язати змінну **_Q_** із виходом **Q:BOOL** екземпляра **_R_TRIG_** функціонального блока **R_TRIG** (див. рис. 20.2).

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Як працює тригер **F_TRIG**?
2. Як працює тригер **R_TRIG**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №21.
# ВИВЧЕННЯ РОБОТИ АНАЛОГОВОГО ПД-РЕГУЛЯТОРА PD, АНАЛОГОВОГО ПІД-РЕГУЛЯТОРА PID І АНАЛОГОВОГО ПІД-РЕГУЛЯТОРА PID_FIXCYCLE

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи аналогового ПД-регулятора **PD**, аналогового ПІД-регулятора **PID** і аналогового ПІД-регулятора **PID_FIXCYCLE**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Формування керуючого сигналу за допомогою функціонального блока аналоговий ПД-регулятор PD (з автоматичним встановленням моментів формування керівного впливу)

#### 2.1.1. Формування керуючого сигналу за допомогою функціонального блока аналоговий ПД-регулятор PD (з автоматичним встановленням моментів формування керівного впливу) з використанням мови програмування Instruction List (IL)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
2. Створити екземпляр **_PD_** функціонального блока **PD**.
3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).
5. Створити змінну **_ACTUAL_** типу **REAL**.
6. Створити змінну **_Y_** типу **REAL**.
7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).
8. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.
9. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.
10. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _ACTUAL_
ST      _PD_.ACTUAL
LD      1.0
ST      _PD_.SET_POINT
LD      5.0
ST      _PD_.KP
LD      0.001
ST      _PD_.TV
LD      1.0
ST      _PD_.Y_MANUAL
```

```
LD      0.0
ST      _PD_.Y_OFFSET
LD      0.0
ST      _PD_.Y_MIN
LD      10.0
ST      _PD_.Y_MAX
LD      _MANUAL_
ST      _PD_.MANUAL
LD      _RESET_
ST      _PD_.RESET
CAL     _PD_
LD      _PD_.Y
ST      _Y_
LD      _PD_.LIMITS_ACTIVE
ST      _LIMITS_ACTIVE_
LD      _Y_
SUB     _ACTUAL_
ST      _INTEGRAL_.IN
LD      10
ST      _INTEGRAL_.TM
LD      _RESET_
ST      _INTEGRAL_.RESET
CAL     _INTEGRAL_
LD      _INTEGRAL_.OUT
ST      _ACTUAL_
LD      _INTEGRAL_.OVERFLOW
ST      _OVERFLOW_INTEGRAL_
```

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN** і **Y_MAX**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретного виходу **DO–1–0** (електромагнітного реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.1.2. Формування керуючого сигналу за допомогою функціонального блока аналоговий ПД-регулятор PD (з автоматичним встановленням моментів формування керівного впливу) з використанням мови програмування Structured Text (ST)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_PD_** функціонального блока **PD**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

9. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.

10. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_PD_(ACTUAL:=_ACTUAL_, SET_POINT:=1.0, KP:=5.0,
TV:=0.001, Y_MANUAL:=1.0, Y_OFFSET:=0.0, Y_MIN:=0.0,
Y_MAX:=10.0, MANUAL:=_MANUAL_, RESET:=_RESET_) ;
_Y_:=_PD_.Y ;
_LIMITS_ACTIVE_ := _PD_.LIMITS_ACTIVE ;
_INTEGRAL_(IN:=_Y_-_ACTUAL_, TM:=10,
RESET:=_RESET_) ;
_ACTUAL_ := _INTEGRAL_.OUT ;
_OVERFLOW_INTEGRAL_ := _INTEGRAL_.OVERFLOW ;
```

11. Запустити цей програмний код на виконання.

12. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN** і **Y_MAX**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретного виходу **DO–1–0** (електромагнітного реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.1.3. Формування керуючого сигналу за допомогою функціонального блока аналоговий ПД-регулятор PD (з автоматичним встановленням моментів формування керівного впливу) з використанням мови програмування Function Block Diagram (FBD)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

2. Створити екземпляр **_PD_** функціонального блока **PD**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Подати на вхід **SET_POINT:REAL** значення **1.0**.

9. Подати на вхід **KP:REAL** значення **5.0**.

10. Подати на вхід **TV:REAL** значення **0.001**.

11. Подати на вхід **Y_MANUAL:REAL** значення **1.0**.

12. Подати на вхід **Y_OFFSET:REAL** значення **0.0**.

13. Подати на вхід **Y_MIN:REAL** значення **0.0**.

14. Подати на вхід **Y_MAX:REAL** значення **10.0**.

15. Програмно зв'язати змінну **_MANUAL_** із входом **MANUAL:BOOL** екземпляра **_PD_** функціонального блока **PD** (див. рис. 21.1).

16. Програмно зв'язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_PD_** функціонального блока **PD** (див. рис. 21.1).

17. Програмно зв'язати змінну **_Y_** із виходом **Y:REAL** екземпляра **_PD_** функціонального блока **PD** (див. рис. 21.1).

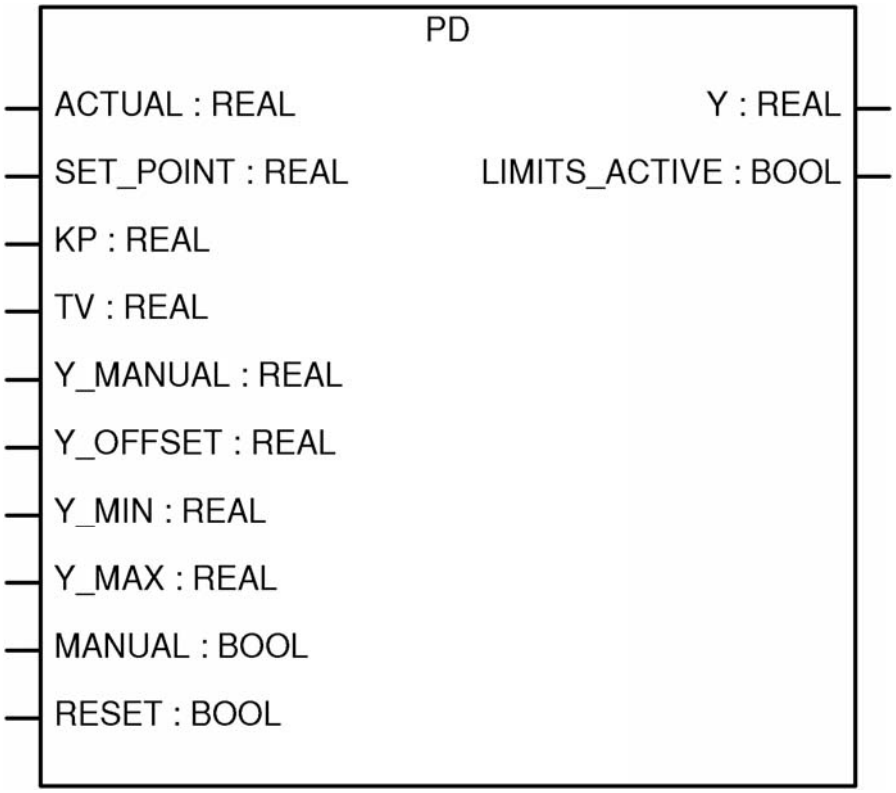



Рис. 21.1. Функціональний блок аналоговий ПД-регулятор **PD** (з автоматичним встановленням моментів формування керівного впливу)

18. Програмно зв'язати змінну **_LIMITS_ACTIVE_** із виходом **LIMITS_ACTIVE:BOOL** екземпляра **_PD_** функціонального блока **PD** (див. рис. 21.1).

19. Створити оператор **SUB**.

20. Програмно зв'язати змінну **_Y_** із першим входом оператора **SUB**.

21. Програмно зв'язати змінну **_ACTUAL_** із другим входом оператора

**SUB**.

22. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

23. Створити змінну **_OVERFLOW_INTEGRAL _** типу **BOOL**.

24. Програмно зв'язати вихід оператора **SUB** із входом **IN:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

25. Подати на вхід **TM:DWORD** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL** значення **10**.

26. Програмно зв'язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

27. Програмно зв'язати змінну **_ACTUAL_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

28. Програмно зв'язати змінну **_ACTUAL_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

29. Програмно зв'язати змінну **_OVERFLOW_INTEGRAL_** із виходом **OVERFLOW:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

30. Запустити цей програмний код на виконання.

31. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN** і **Y_MAX**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретного виходу **DO–1–0** (електромагнітного реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

## 2.2. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID (з автоматичним встановленням моментів формування керівного впливу)

### 2.2.1. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID (з автоматичним встановленням моментів формування керівного впливу) з використанням мови програмування Instruction List (IL)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

2. Створити екземпляр **_PID_** функціонального блока **PID**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0**

(яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.1** (яка відповідає дискретному виходу **DO–2–0**).

9. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

10. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.

11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _ACTUAL_
ST      _PID_.ACTUAL
LD      1.0
ST      _PID_.SET_POINT
LD      5.0
ST      _PID_.KP
LD      10.0
ST      _PID_.TN
LD      0.001
ST      _PID_.TV
LD      1.0
ST      _PID_.Y_MANUAL
LD      0.0
ST      _PID_.Y_OFFSET
LD      0.0
ST      _PID_.Y_MIN
LD      10.0
ST      _PID_.Y_MAX
LD      _MANUAL_
ST      _PID_.MANUAL
LD      _RESET_
ST      _PID_.RESET
CAL     _PID_
LD      _PID_.Y
ST      _Y_
LD      _PID_.LIMITS_ACTIVE
ST      _LIMITS_ACTIVE_
LD      _PID_.OVERFLOW
ST      _OVERFLOW_
LD      _Y_
SUB     _ACTUAL_
ST      _INTEGRAL_.IN
```

```
LD          10
ST          _INTEGRAL_.TM
LD          _RESET_
ST          _INTEGRAL_.RESET
CAL         _INTEGRAL_
LD          _INTEGRAL_.OUT
ST          _ACTUAL_
LD          _INTEGRAL_.OVERFLOW
ST          _OVERFLOW_INTEGRAL_
```

12. Запустити цей програмний код на виконання.

13. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TN**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN** і **Y_MAX**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.2.2. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID (з автоматичним встановленням моментів формування керівного впливу) з використанням мови програмування Structured Text (ST)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_PID_** функціонального блока **PID**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.1** (яка відповідає дискретному виходу **DO–2–0**).

9. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

10. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.

11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_PID_(ACTUAL:=_ACTUAL_, SET_POINT:=1.0, KP:=5.0,
TN:= 0.001, TV:=0.001, Y_MANUAL:=1.0, Y_OFFSET:=0.0,
Y_MIN:=0.0, Y_MAX:=10.0, MANUAL:=_MANUAL_,
```

**RESET:=_RESET_) ;**
**_Y_:=_PID_.Y ;**
**_LIMITS_ACTIVE_ := _PID_.LIMITS_ACTIVE ;**
**_OVERFLOW_ := _PID_.OVERFLOW ;**
**_INTEGRAL_(IN:=_Y_-_ACTUAL_, TM:=10,**
**RESET:=_RESET_) ;**
**_ACTUAL_ := _INTEGRAL_.OUT ;**
**_OVERFLOW_INTEGRAL_ := _INTEGRAL_.OVERFLOW ;**

12. Запустити цей програмний код на виконання.

13. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TN**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN** і **Y_MAX**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.2.3. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID (з автоматичним встановленням моментів формування керівного впливу) з використанням мови програмування Function Block Diagram (FBD)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

2. Створити екземпляр **_PID_** функціонального блока **PID**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.1** (яка відповідає дискретному виходу **DO–2–0**).

9. Подати на вхід **SET_POINT:REAL** значення **1.0**.

10. Подати на вхід **KP:REAL** значення **5.0**.

11. Подати на вхід **TN:REAL** значення **10.0**.

12. Подати на вхід **TV:REAL** значення **0.001**.

13. Подати на вхід **Y_MANUAL:REAL** значення **1.0**.

14. Подати на вхід **Y_OFFSET:REAL** значення **0.0**.

15. Подати на вхід **Y_MIN:REAL** значення **0.0**.

16. Подати на вхід **Y_MAX:REAL** значення **10.0**.

17. Програмно зв’язати змінну **_MANUAL_** із входом **MANUAL:BOOL**

екземпляра **_PID_** функціонального блока **PID** (див. рис. 21.2).

18. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_PID_** функціонального блока **PID** (див. рис. 21.2).

19. Програмно зв’язати змінну **_Y_** із виходом **Y:REAL** екземпляра **_PID_** функціонального блока **PID** (див. рис. 21.2).

20. Програмно зв’язати змінну **_LIMITS_ACTIVE_** із виходом **LIMITS_ACTIVE:BOOL** екземпляра **_PID_** функціонального блока **PID** (див. рис. 21.2).

21. Програмно зв’язати змінну **_OVERFLOW_** із виходом **OVERFLOW:BOOL** екземпляра **_PID_** функціонального блока **PID** (див. рис. 21.2).

22. Створити оператор **SUB**.

23. Програмно зв’язати змінну **_Y_** із першим входом оператора **SUB**.

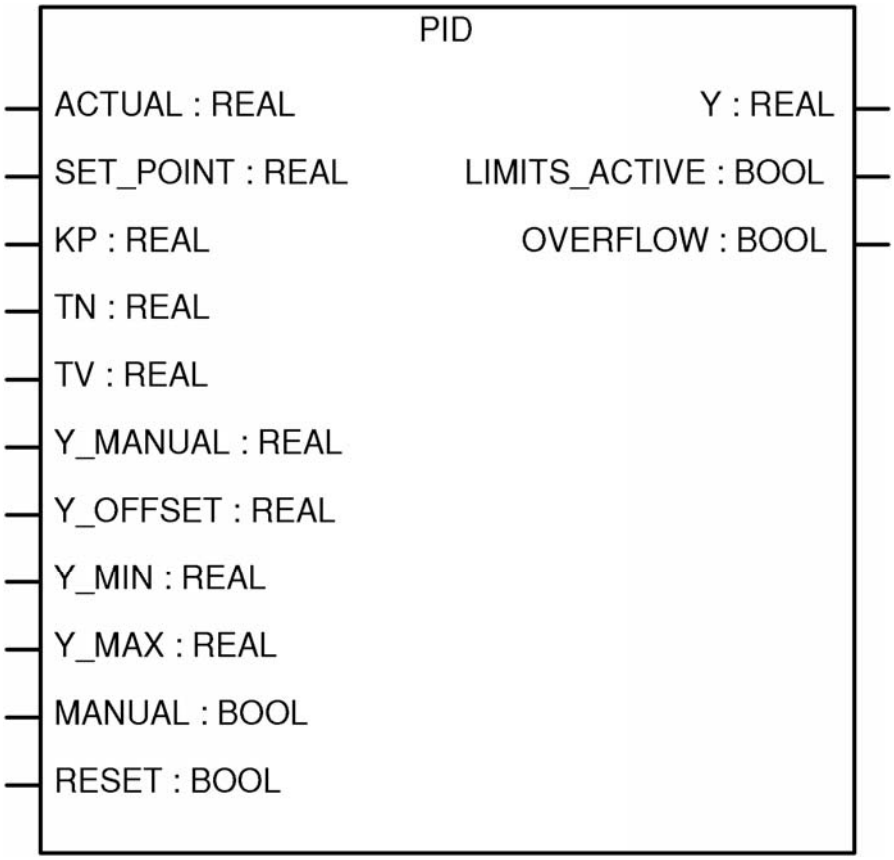



Рис. 21.2. Функціональний блок аналоговий ПІД-регулятор **PID** (з автоматичним встановленням моментів формування керівного впливу)

24. Програмно зв’язати змінну **_ACTUAL_** із другим входом оператора **SUB**.

25. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

26. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.

27. Програмно зв’язати вихід оператора **SUB** із входом **IN:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

28. Подати на вхід **TM:DWORD** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL** значення **10**.

29. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

30. Програмно зв’язати змінну **_ACTUAL_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

31. Програмно зв’язати змінну **_ACTUAL_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

32. Програмно зв’язати змінну **_OVERFLOW_INTEGRAL_** із виходом **OVERFLOW:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

33. Запустити цей програмний код на виконання.

34. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TN**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN** і **Y_MAX**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

## 2.3. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID_FIXCYCLE (з ручним встановленням моментів формування керівного впливу)

### 2.3.1. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID_FIXCYCLE (з ручним встановленням моментів формування керівного впливу) з використанням мови програмування Instruction List (IL)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

2. Створити екземпляр **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.1** (яка відповідає дискретному виходу **DO–2–0**).

9. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

10. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.
11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _ACTUAL_
ST      _PID_FIXCYCLE_.ACTUAL
LD      1.0
ST      _PID_FIXCYCLE_.SET_POINT
LD      5.0
ST      _PID_FIXCYCLE_.KP
LD      10.0
ST      _PID_FIXCYCLE_.TN
LD      0.001
ST      _PID_FIXCYCLE_.TV
LD      1.0
ST      _PID_FIXCYCLE_.Y_MANUAL
LD      0.0
ST      _PID_FIXCYCLE_.Y_OFFSET
LD      0.0
ST      _PID_FIXCYCLE_.Y_MIN
LD      10.0
ST      _PID_FIXCYCLE_.Y_MAX
LD      _MANUAL_
ST      _PID_FIXCYCLE_.MANUAL
LD      _RESET_
ST      _PID_FIXCYCLE_.RESET
LD      0.1
ST      _PID_FIXCYCLE_.CYCLE
CAL     _PID_FIXCYCLE_
LD      _PID_FIXCYCLE_.Y
ST      _Y_
LD      _PID_FIXCYCLE_.LIMITS_ACTIVE
ST      _LIMITS_ACTIVE_
LD      _PID_FIXCYCLE_.OVERFLOW
ST      _OVERFLOW_
LD      _Y_
SUB     _ACTUAL_
ST      _INTEGRAL_.IN
LD      10
ST      _INTEGRAL_.TM
```

```
LD          _RESET_
ST          _INTEGRAL_.RESET
CAL         _INTEGRAL_
LD          _INTEGRAL_.OUT
ST          _ACTUAL_
LD          _INTEGRAL_.OVERFLOW
ST          _OVERFLOW_INTEGRAL_
```

12. Запустити цей програмний код на виконання.

13. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TN**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN**, **Y_MAX** і **CYCLE**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.3.2. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID_FIXCYCLE (з ручним встановленням моментів формування керівного впливу) з використанням мови програмування Structured Text (ST)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.1** (яка відповідає дискретному виходу **DO–2–0**).

9. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

10. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.

11. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_PID_FIXCYCLE_(ACTUAL:=_ACTUAL_, SET_POINT:=1.0,
KP:=5.0, TN:= 0.001, TV:=0.001, Y_MANUAL:=1.0,
Y_OFFSET:=0.0, Y_MIN:=0.0, Y_MAX:=10.0,
MANUAL:=_MANUAL_, RESET:=_RESET_, CYCLE:=0.1) ;
```

```
_Y_:=_PID_FIXCYCLE_.Y ;
_LIMITS_ACTIVE_ := _PID_FIXCYCLE_.LIMITS_ACTIVE ;
_OVERFLOW_ := _PID_FIXCYCLE_.OVERFLOW ;
_INTEGRAL_(IN:=_Y_-_ACTUAL_,                    TM:=10,
RESET:=_RESET_) ;
_ACTUAL_ := _INTEGRAL_.OUT ;
_OVERFLOW_INTEGRAL_ := _INTEGRAL_.OVERFLOW ;
```

12. Запустити цей програмний код на виконання.

13. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TN**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN**, **Y_MAX** і **CYCLE**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.3.3. Формування керуючого сигналу за допомогою аналогового ПІД-регулятора PID_FIXCYCLE (з ручним встановленням моментів формування керівного впливу) з використанням мови програмування Function Block Diagram (FBD)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

2. Створити екземпляр **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE**.

3. Створити змінну **_MANUAL_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

5. Створити змінну **_ACTUAL_** типу **REAL**.

6. Створити змінну **_Y_** типу **REAL**.

7. Створити змінну **_LIMITS_ACTIVE_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.1** (яка відповідає дискретному виходу **DO–2–0**).

9. Подати на вхід **SET_POINT:REAL** значення **1.0**.

10. Подати на вхід **KP:REAL** значення **5.0**.

11. Подати на вхід **TN:REAL** значення **10.0**.

12. Подати на вхід **TV:REAL** значення **0.001**.

13. Подати на вхід **Y_MANUAL:REAL** значення **1.0**.

14. Подати на вхід **Y_OFFSET:REAL** значення **0.0**.

15. Подати на вхід **Y_MIN:REAL** значення **0.0**.

16. Подати на вхід **Y_MAX:REAL** значення **10.0**.

17. Подати на вхід **CYCLE:REAL** значення **0.1**.

18. Програмно зв’язати змінну **_MANUAL_** із входом **MANUAL:BOOL** екземпляра **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE** (див. рис. 21.3).

19. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE** (див. рис. 21.3).

20. Програмно зв’язати змінну **_Y_** із виходом **Y:REAL** екземпляра **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE** (див. рис. 21.3).

20. Програмно зв’язати змінну **_LIMITS_ACTIVE_** із виходом **LIMITS_ACTIVE:BOOL** екземпляра **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE** (див. рис. 21.3).

21. Програмно зв’язати змінну **_OVERFLOW_** із виходом **OVERFLOW:BOOL** екземпляра **_PID_FIXCYCLE_** функціонального блока **PID_FIXCYCLE** (див. рис. 21.3).

22. Створити оператор **SUB**.

23. Програмно зв’язати змінну **_Y_** із першим входом оператора **SUB**.

24. Програмно зв’язати змінну **_ACTUAL_** із другим входом оператора **SUB**.

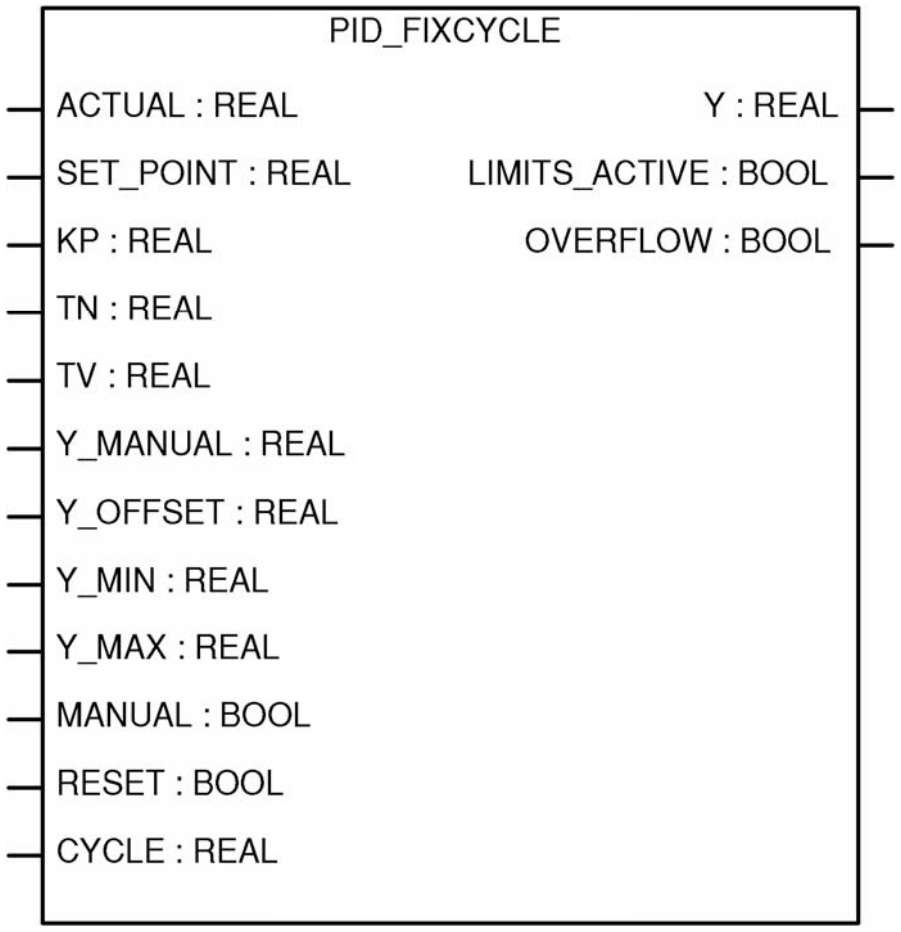



Рис. 21.3. Функціональний блок аналоговий ПІД-регулятор **PID_FIXCYCLE** (з ручним встановленням моментів формування керівного впливу)

25. Створити екземпляр **_INTEGRAL_** функціонального блока

**INTEGRAL**.

26. Створити змінну **_OVERFLOW_INTEGRAL_** типу **BOOL**.

27. Програмно зв’язати вихід оператора **SUB** із входом **IN:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

28. Подати на вхід **TM:DWORD** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL** значення **10**.

29. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

30. Програмно зв’язати змінну **_ACTUAL_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

31. Програмно зв’язати змінну **_ACTUAL_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

32. Програмно зв’язати змінну **_OVERFLOW_INTEGRAL_** із виходом **OVERFLOW:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL**.

33. Запустити цей програмний код на виконання.

34. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **SET_POINT**, **KP**, **TN**, **TV**, **Y_MANUAL**, **Y_OFFSET**, **Y_MIN**, **Y_MAX** і **CYCLE**, виконати вимірювання значень змінних **_ACTUAL_** і **_Y_**, а також стану дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле). Значення змінних **_ACTUAL_** і **_Y_**, а також стан дискретних виходів **DO–1–0** і **DO–2–0** (електромагнітних реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 3. Питання для самоперевірки

1. Які регулятори найчастіше використовуються для автоматизації технологічних процесів та виробництв?
2. Яка існує різниця між аналоговими (неперервними) і дискретними (цифровими) регуляторами?
3. Як в процесі керування використовується пропорційна складова ПІД-регулятора?
4. Як в процесі керування використовується диференціююча складова ПІД-регулятора?
5. Як в процесі керування використовується інтегруюча складова ПІД-регулятора?
6. Як працює аналоговий ПД-регулятор **PD**?
7. Як працює аналоговий ПІД-регулятор **PID**?
8. Як працює аналоговий ПІД-регулятор **PID_FIXCYCLE**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №22.
# ВИВЧЕННЯ РОБОТИ КУСКОВО-ЛІНІЙНОЇ ІНТЕРПОЛЯЦІЇ СИГНАЛУ CHARCURVE, ОБМЕЖЕННЯ ШВИДКОСТІ ЗМІНИ СИГНАЛУ RAMP_INT І ОБМЕЖЕННЯ ШВИДКОСТІ ЗМІНИ СИГНАЛУ RAMP_REAL

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи кусково-лінійної інтерполяції сигналу **CHARCURVE**, обмеження швидкості зміни сигналу **RAMP_INT** (цілочисловий опис) і обмеження швидкості зміни сигналу **RAMP_REAL** (дійсний опис).

## 2. Порядок виконання лабораторної роботи

### 2.1. Кусково-лінійна інтерполяція сигналу за допомогою функціонального блока кусково-лінійний інтерполятор CHARCURVE

#### 2.1.1. Кусково-лінійна інтерполяція сигналу за допомогою функціонального блока кусково-лінійний інтерполятор CHARCURVE з використанням мови програмування Instruction List (IL)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
2. Створити екземпляр **_CHARCURVE_** функціонального блока **CHARCURVE**.
3. Створити змінну **_P_** типу **ARRAY [0..10] OF POINT** з початковими значеннями **(0, 0)**, **(10, 10)**, **(20, 40)**, **(30, 90)**, **(40, 160)**, **(50, 250)**, (**60, 360)**, **(70, 490)**, **(80, 640)**, **(90, 810)** і **(100, 1000)**.
4. Створити змінну **_OUT_** типу **INT**.
5. Створити змінну **_ERR_** типу **BYTE**.
6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      5
ST      _CHARCURVE_.IN
LD      11
ST      _CHARCURVE_.N
CAL     _CHARCURVE_(P:=_P_)
LD      _CHARCURVE_.OUT
ST      _OUT_
LD      _CHARCURVE_.ERR
ST      _ERR_
```

7. Запустити цей програмний код на виконання.
8. Змінюючи значення, які подаються на входи **IN**, **N** і **P**, виконати вимірювання значень змінних **_OUT_**, **_ERR_** і **_P_**. Значення змінних **_OUT_**, **_ERR_** і **_P_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.2. Кусково-лінійна інтерполяція сигналу за допомогою функціонального блока кусково-лінійний інтерполятор CHARCURVE з використанням мови програмування Structured Text (ST)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_CHARCURVE_** функціонального блока **CHARCURVE**.

3. Створити змінну **_P_** типу **ARRAY [0..10] OF POINT** з початковими значеннями **(0, 0)**, **(10, 10)**, **(20, 40)**, **(30, 90)**, **(40, 160)**, **(50, 250)**, (**60, 360)**, **(70, 490)**, **(80, 640)**, **(90, 810)** і **(100, 1000)**.

4. Створити змінну **_OUT_** типу **INT**.

5. Створити змінну **_ERR_** типу **BYTE**.

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_CHARCURVE_(IN:=5, N:=11, P:=_P_) ;
_OUT_ := _CHARCURVE_.OUT ;
_ERR_ := _CHARCURVE_.ERR ;
```

7. Запустити цей програмний код на виконання.

8. Змінюючи значення, які подаються на входи **IN**, **N** і **P**, виконати вимірювання значень змінних **_OUT_**, **_ERR_** і **_P_**. Значення змінних **_OUT_**, **_ERR_** і **_P_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.3. Кусково-лінійна інтерполяція сигналу за допомогою функціонального блока кусково-лінійний інтерполятор CHARCURVE з використанням мови програмування Function Block Diagram (FBD)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

2. Створити екземпляр **_CHARCURVE_** функціонального блока **CHARCURVE**.

3. Створити змінну **_P_** типу **ARRAY [0..10] OF POINT** з початковими значеннями **(0, 0)**, **(10, 10)**, **(20, 40)**, **(30, 90)**, **(40, 160)**, **(50, 250)**, (**60, 360)**, **(70, 490)**, **(80, 640)**, **(90, 810)** і **(100, 1000)**.

4. Створити змінну **_OUT_** типу **INT**.

5. Створити змінну **_ERR_** типу **BYTE**.

6. Подати на вхід **IN:INT** значення **5**.

7. Подати на вхід **N:BYTE** значення **11**.

8. Програмно зв’язати змінну **_P_** із входом **P:ARRAY [0..10] OF POINT (VAR_IN_OUT)** екземпляра **_CHARCURVE_** функціонального блока **CHARCURVE** (див. рис. 22.1).

9. Програмно зв’язати змінну **_OUT_** із виходом **OUT:INT** екземпляра **_CHARCURVE_** функціонального блока **CHARCURVE** (див. рис. 22.1).

10. Програмно зв’язати змінну **_ERR_** із виходом **ERR:BYTE** екземпляра **_CHARCURVE_** функціонального блока **CHARCURVE** (див. рис. 22.1).

11. Програмно зв’язати змінну **_P_** із виходом **P:ARRAY [0..10] OF POINT (VAR_IN_OUT)** екземпляра **_CHARCURVE_** функціонального блока **CHARCURVE** (див. рис. 22.1).

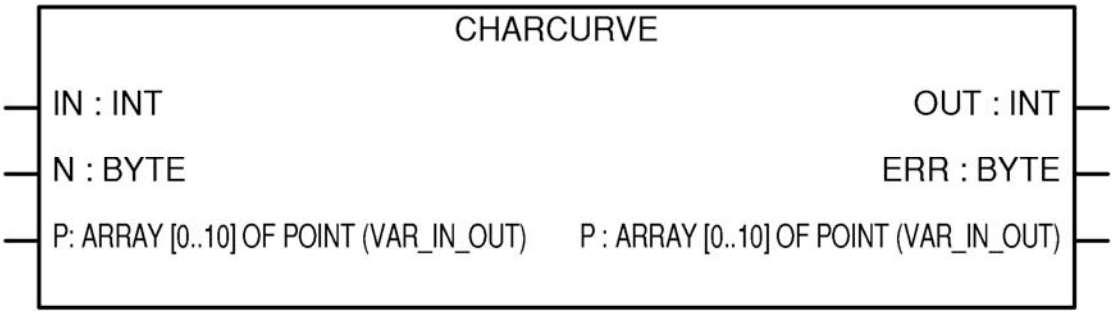



Рис. 22.1. Функціональний блок кусково-лінійний інтерполятор **CHARCURVE**

12. Запустити цей програмний код на виконання.

13. Змінюючи значення, які подаються на входи **IN**, **N** і **P**, виконати вимірювання значень змінних **_OUT_**, **_ERR_** і **_P_**. Значення змінних **_OUT_**, **_ERR_** і **_P_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_INT (цілочисловий опис)

#### 2.2.1. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_INT (цілочисловий опис) з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_RAMP_INT_** функціонального блока **RAMP_INT**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **INT**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
        LD      _i_
        MOD     200
        LT      100
        JMPC    M1
        LD      -100
        ST      _RAMP_INT_.IN
        JMP     M2
M1:     LD      100
        ST      _RAMP_INT_.IN
M2:     LD      2
        ST      _RAMP_INT_.ASCEND
        LD      2
```

```
ST          _RAMP_INT_.DESCEND
LD          T#1s
ST          _RAMP_INT_.TIMEBASE
LD          _RESET_
ST          _RAMP_INT_.RESET
CAL         _RAMP_INT_
LD          _RAMP_INT_.OUT
ST          _OUT_
LD          _i_
ADD         1
ST          _i_
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN**, **ASCEND**, **DESCEND** і **TIMEBASE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.2.2. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_INT (цілочисловий опис) з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

3. Створити екземпляр **_RAMP_INT_** функціонального блока **RAMP_INT**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **INT**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
IF _i_ MOD 200 < 100 THEN
        _RAMP_INT_.IN := 100 ;
ELSE
        _RAMP_INT_.IN := –100 ;
END_IF
_RAMP_INT_(ASCEND:=2, DESCEND:=2, TIMEBASE:=T#1s,
RESET:=_RESET_) ;
_OUT_ := _RAMP_INT_.OUT ;
_i_ := _i_+1 ;
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN**, **ASCEND**, **DESCEND** і **TIMEBASE**, виконати вимірювання

значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.2.3. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_INT (цілочисловий опис) з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

3. Створити екземпляр **_RAMP_INT_** функціонального блока **RAMP_INT**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_IN_** типу **INT**.

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

7. Створити змінну **_OUT_** типу **INT**.

8. Програмно зв’язати змінну **_IN_** із входом **IN:INT** екземпляра **_RAMP_INT_** функціонального блока **RAMP_INT** (див. рис. 22.2).

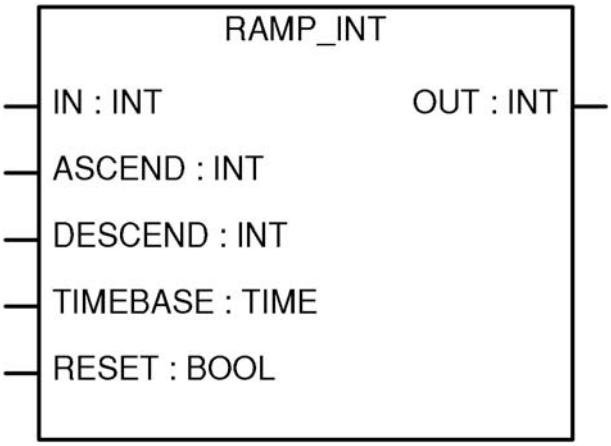



Рис. 22.2. Функціональний блок обмежувач швидкості зміни сигналу **RAMP_INT** (цілочисловий опис)

9. Подати на вхід **ASCEND:INT** значення **2**.

10. Подати на вхід **DESCEND:INT** значення **2**.

11. Подати на вхід **TIMEBASE:TIME** значення **T#1s**.

12. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_RAMP_INT_** функціонального блока **RAMP_INT** (див. рис. 22.2).

13. Програмно зв’язати змінну **_OUT_** із виходом **OUT:INT** екземпляра **_RAMP_INT_** функціонального блока **RAMP_INT** (див. рис. 22.2).

14. Створити оператор **MOD**.

15. Програмно зв’язати змінну **_i_** із першим входом оператора **MOD**.

16. Подати на другий вхід оператора **MOD** значення **200**.

17. Створити оператор **LT**.

18. Подати на перший вхід оператора **LT** значення з виходу оператора **MOD**.

19. Подати на другий вхід оператора **LT** значення **100**.

20. Створити оператор **SEL**.

21. Подати на перший вхід оператора **SEL** значення з виходу оператора **LT**.

22. Подати на другий вхід оператора **SEL** значення **–100**.

23. Подати на третій вхід оператора **SEL** значення **100**.

34. Програмно зв'язати змінну **_IN_** із виходом оператора **SEL**.

35. Створити оператор **ADD**.

36. Програмно зв'язати змінну **_i_** із першим входом оператора **ADD**.

37. Подати на другий вхід оператора **ADD** значення **1**.

38. Програмно зв'язати змінну **_i_** із виходом оператора **ADD**.

39. Запустити цей програмний код на виконання.

40. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN**, **ASCEND**, **DESCEND** і **TIMEBASE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.3. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_REAL (дійсний опис)

#### 2.3.1. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_REAL (дійсний опис) з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_RAMP_REAL_** функціонального блока **RAMP_REAL**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **REAL**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _i_
MOD     200
LT      100
JMPC    M1
LD      -100.0
ST      _RAMP_REAL_.IN
JMP     M2
```

```
M1:     LD      100.0
        ST      _RAMP_REAL_.IN
M2:     LD      2.0
        ST      _RAMP_REAL_.ASCEND
        LD      2.0
        ST      _RAMP_REAL_.DESCEND
        LD      T#1s
        ST      _RAMP_REAL_.TIMEBASE
        LD      _RESET_
        ST      _RAMP_REAL_.RESET
        CAL     _RAMP_REAL_
        LD      _RAMP_REAL_.OUT
        ST      _OUT_
        LD      _i_
        ADD     1
        ST      _i_
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN**, **ASCEND**, **DESCEND** і **TIMEBASE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.3.2. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_REAL (дійсний опис) з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

3. Створити екземпляр **_RAMP_REAL_** функціонального блока **RAMP_REAL**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **REAL**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
IF _i_ MOD 200 < 100 THEN
        _RAMP_REAL_.IN := 100.0 ;
ELSE
        _RAMP_REAL_.IN := –100.0 ;
END_IF
_RAMP_REAL_(ASCEND:=2.0,                DESCEND:=2.0,
TIMEBASE:=T#1s, RESET:=_RESET_) ;
_OUT_ := _RAMP_REAL_.OUT ;
```

**_i_ := _i_+1 ;**

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN**, **ASCEND**, **DESCEND** і **TIMEBASE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.3.3. Обмеження швидкості зміни сигналу за допомогою функціонального блока обмежувач швидкості зміни сигналу RAMP_REAL (дійсний опис) з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

3. Створити екземпляр **_RAMP_REAL_** функціонального блока **RAMP_REAL**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_IN_** типу **REAL**.

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

7. Створити змінну **_OUT_** типу **REAL**.

8. Програмно зв’язати змінну **_IN_** із входом **IN:REAL** екземпляра **_RAMP_REAL_** функціонального блока **RAMP_REAL** (див. рис. 22.3).

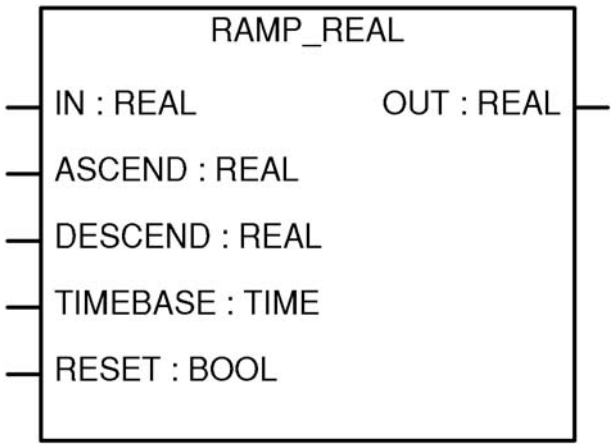



Рис. 22.3. Функціональний блок обмежувач швидкості зміни сигналу **RAMP_REAL** (дійсний опис)

9. Подати на вхід **ASCEND:REAL** значення **2.0**.

10. Подати на вхід **DESCEND:REAL** значення **2.0**.

11. Подати на вхід **TIMEBASE:TIME** значення **T#1s**.

12. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_RAMP_REAL_** функціонального блока **RAMP_REAL** (див. рис. 22.3).

13. Програмно зв’язати змінну **_OUT_** із виходом **OUT:REAL** екземпляра **_RAMP_REAL_** функціонального блока **RAMP_REAL** (див. рис. 22.3).

14. Створити оператор **MOD**.

15. Програмно зв’язати змінну **_i_** із першим входом оператора **MOD**.

16. Подати на другий вхід оператора **MOD** значення **200**.

17. Створити оператор **LT**.

18. Подати на перший вхід оператора **LT** значення з виходу оператора **MOD**.

19. Подати на другий вхід оператора **LT** значення **100**.

20. Створити оператор **SEL**.

21. Подати на перший вхід оператора **SEL** значення з виходу оператора **LT**.

22. Подати на другий вхід оператора **SEL** значення **–100.0**.

23. Подати на третій вхід оператора **SEL** значення **100.0**.

34. Програмно зв’язати змінну **_IN_** із виходом оператора **SEL**.

35. Створити оператор **ADD**.

36. Програмно зв’язати змінну **_i_** із першим входом оператора **ADD**.

37. Подати на другий вхід оператора **ADD** значення **1**.

38. Програмно зв’язати змінну **_i_** із виходом оператора **ADD**.

39. Запустити цей програмний код на виконання.

40. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN**, **ASCEND**, **DESCEND** і **TIMEBASE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 3. Питання для самоперевірки

1. Яка існує відмінність між інтерполяцією, екстраполяцією і апроксимацією?

2. Які алгоритми використовуються для здійснення лінійної інтерполяції?

3. Які алгоритми використовуються для здійснення квадратичної інтерполяції?

4. Які алгоритми використовуються для здійснення кубічної інтерполяції?

5. Як працює кусково-лінійний інтерполятор **CHARCURVE**?

6. Які алгоритми використовуються для обмеження швидкості зміни сигналу?

7. Як працює обмежувач швидкості зміни сигналу **RAMP_INT** (цілочисловий опис)?

8. Як працює обмежувач швидкості зміни сигналу **RAMP_REAL** (дійсний опис)?

### 4. Рекомендована література

1. Основна література [1о–8о].

2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №23. ВИВЧЕННЯ РОБОТИ ДИФЕРЕНЦІЮВАННЯ СИГНАЛУ DERIVATIVE, ІНТЕГРУВАННЯ СИГНАЛУ INTEGRAL, ПЕРЕТВОРЕННЯ ДІАПАЗОНІВ СИГНАЛУ LIN_TRAFO, ВИЗНАЧЕННЯ ЗНАЧЕНЬ СИГНАЛУ STATISTICS_INT, ВИЗНАЧЕННЯ ЗНАЧЕНЬ СИГНАЛУ STATISTICS_REAL І ВИЗНАЧЕННЯ ДИСПЕРСІЇ СИГНАЛУ VARIANCE

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи диференціювання сигналу **DERIVATIVE**, інтегрування сигналу **INTEGRAL**, перетворення діапазонів сигналу **LIN_TRAFO**, визначення значень сигналу **STATISTICS_INT** (цілочисловий опис), визначення значень сигналу **STATISTICS_REAL** (дійсний опис) і визначення дисперсії сигналу **VARIANCE**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Диференціювання сигналу за допомогою функціонального блока диференціатор сигналу DERIVATIVE

#### 2.1.1. Диференціювання сигналу за допомогою функціонального блока диференціатор сигналу DERIVATIVE з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
3. Створити екземпляр **_DERIVATIVE_** функціонального блока **DERIVATIVE**.
4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.
5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
6. Створити змінну **_OUT_** типу **REAL**.
7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      0.0314159265
MUL     _i_
SIN
ST      _DERIVATIVE_.IN
LD      1
ST      _DERIVATIVE_.TM
LD      _RESET_
ST      _DERIVATIVE_.RESET
CAL     _DERIVATIVE_
LD      _DERIVATIVE_.OUT
```

```
ST      _OUT_
LD      _i_
ADD     1
ST      _i_
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN** і **TM**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.1.2. Диференціювання сигналу за допомогою функціонального блока диференціатор сигналу DERIVATIVE з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_DERIVATIVE_** функціонального блока **DERIVATIVE**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **REAL**.

7. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_DERIVATIVE_(IN:=SIN(0.0314159265*_i_),          TM:=1,
RESET:=_RESET_) ;

_OUT_ := _DERIVATIVE_.OUT ;

_i_ := _i_+1 ;
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN** і **TM**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.1.3. Диференціювання сигналу за допомогою функціонального блока диференціатор сигналу DERIVATIVE з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

3. Створити екземпляр **_DERIVATIVE_** функціонального блока **DERIVATIVE**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **REAL**.

7. Подати на вхід **IN:REAL** значення **SIN(0.0314159265*_i_)**.

8. Подати на вхід **TM:DWORD** значення **1**.

9. Програмно зв'язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_DERIVATIVE_** функціонального блока **DERIVATIVE** (див. рис. 23.1).

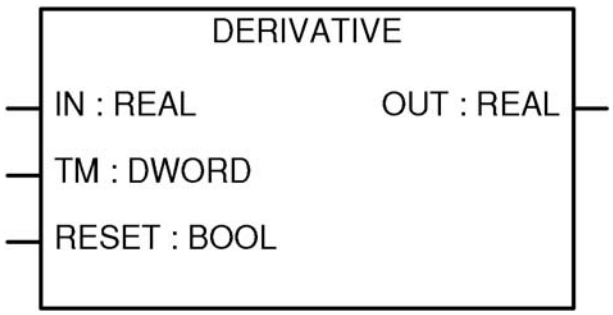



Рис. 23.1. Функціональний блок диференціатор сигналу **DERIVATIVE**

10. Програмно зв'язати змінну **_OUT_** із виходом **OUT:REAL** екземпляра **_DERIVATIVE_** функціонального блока **DERIVATIVE** (див. рис. 23.1).

11. Створити оператор **ADD**.

12. Програмно зв'язати змінну **_i_** із першим входом оператора **ADD**.

13. Подати на другий вхід оператора **ADD** значення **1**.

14. Програмно зв'язати змінну **_i_** із виходом оператора **ADD**.

15. Запустити цей програмний код на виконання.

16. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN** і **TM**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.2. Інтегрування сигналу за допомогою функціонального блока інтегратор сигналу INTEGRAL

#### 2.2.1. Інтегрування сигналу за допомогою функціонального блока інтегратор сигналу INTEGRAL з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відпо-

відає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **REAL**.

7. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      0.0314159265
MUL     _i_
SIN
ST      _INTEGRAL_.IN
LD      1
ST      _INTEGRAL_.TM
LD      _RESET_
ST      _INTEGRAL_.RESET
CAL     _INTEGRAL_
LD      _INTEGRAL_.OUT
ST      _OUT_
LD      _INTEGRAL_.OVERFLOW
ST      _OVERFLOW_
LD      _i_
ADD     1
ST      _i_
```

8. Запустити цей програмний код на виконання.

9. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN** і **TM**, виконати вимірювання значення змінної **_OUT_**, а також стану дискретного виходу **DO–1–0 (**електромагнітного реле). Значення змінної **_OUT_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.2.2. Інтегрування сигналу за допомогою функціонального блока інтегратор сигналу INTEGRAL з використанням мови програмування Structured Text (ST)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_OUT_** типу **REAL**.

7. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_INTEGRAL_(IN:=SIN(0.0314159265*_i_),          TM:=1,
RESET:=_RESET_) ;
_OUT_ := _INTEGRAL_.OUT ;
_OVERFLOW_ := _INTEGRAL_.OVERFLOW ;
_i_ := _i_+1 ;
```

9. Запустити цей програмний код на виконання.

10. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN** і **TM**, виконати вимірювання значення змінної **_OUT_**, а також стану дискретного виходу **DO–1–0** (електромагнітного реле). Значення змінної **_OUT_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.2.3. Інтегрування сигналу за допомогою функціонального блока інтегратор сигналу INTEGRAL з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_INTEGRAL_** функціонального блока **INTEGRAL**.

4. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_OUT_** типу **REAL**.

6. Створити змінну **_OVERFLOW_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

7. Подати на вхід **IN:REAL** значення **SIN(0.0314159265*_i_)**.

8. Подати на вхід **TM:DWORD** значення **1**.

9. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL** (див. рис. 23.2).

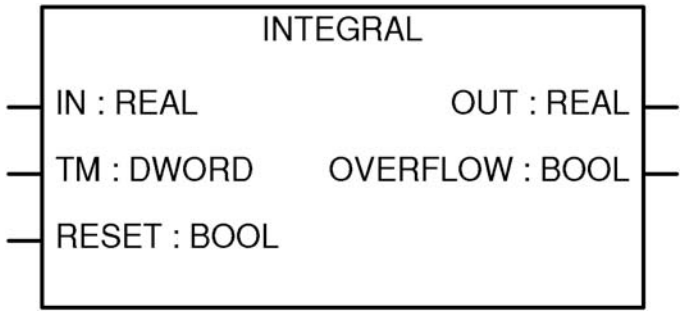



Рис. 23.2. Функціональний блок інтегратор сигналу **INTEGRAL**

10. Програмно зв’язати змінну **_OUT_** із виходом **OUT:REAL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL** (див. рис. 23.2).

11. Програмно зв’язати змінну **_OVERFLOW_** із виходом **OVERFLOW:BOOL** екземпляра **_INTEGRAL_** функціонального блока **INTEGRAL** (див. рис. 23.2).

12. Створити оператор **ADD**.

13. Програмно зв’язати змінну **_i_** із першим входом оператора **ADD**.

14. Подати на другий вхід оператора **ADD** значення **1**.

15. Програмно зв’язати змінну **_i_** із виходом оператора **ADD**.

16. Запустити цей програмний код на виконання.

17. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **IN** і **TM**, виконати вимірювання значення змінної **_OUT_**, а також стану дискретного виходу **DO–1–0** (електромагнітного реле). Значення змінної **_OUT_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 2.3. Перетворення діапазонів сигналу за допомогою функціонального блока перетворювач діапазонів сигналу LIN_TRAFO

#### 2.3.1. Перетворення діапазонів сигналу за допомогою функціонального блока перетворювач діапазонів сигналу LIN_TRAFO з використанням мови програмування Instruction List (IL)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

2. Створити екземпляр **_LIN_TRAFO_** функціонального блока **LIN_TRAFO**.

3. Створити змінну **_OUT_** типу **REAL**.

4. Створити змінну **_ERROR_** типу **BOOL**.

5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      0.5
ST      _LIN_TRAFO_.IN
LD      0.0
ST      _LIN_TRAFO_.IN_MIN
LD      1.0
ST      _LIN_TRAFO_.IN_MAX
LD      0.0
ST      _LIN_TRAFO_.OUT_MIN
LD      10.0
ST      _LIN_TRAFO_.OUT_MAX
CAL     _LIN_TRAFO_
LD      _LIN_TRAFO_.OUT
ST      _OUT_
```

```
LD          _LIN_TRAFO_.ERROR
ST          _ERROR_
```

6. Запустити цей програмний код на виконання.

7. Змінюючи значення, які подаються на входи **IN**, **IN_MIN**, **IN_MAX**, **OUT_MIN** і **OUT_MAX**, виконати вимірювання значень змінних **_OUT_** і **_ERROR_**. Значення змінних **_OUT_** і **_ERROR_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.2. Перетворення діапазонів сигналу за допомогою функціонального блока перетворювач діапазонів сигналу LIN_TRAFO з використанням мови програмування Structured Text (ST)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_LIN_TRAFO_** функціонального блока **LIN_TRAFO**.

3. Створити змінну **_OUT_** типу **REAL**.

4. Створити змінну **_ERROR_** типу **BOOL**.

5. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_LIN_TRAFO_(IN:=0.5,  IN_MIN:=0.0,  IN_MAX:=1.0,
OUT_MIN:=0.0, OUT_MAX:=10.0) ;
_OUT_ := _LIN_TRAFO_.OUT ;
_ERROR_ := _LIN_TRAFO_.ERROR ;
```

6. Запустити цей програмний код на виконання.

7. Змінюючи значення, які подаються на входи **IN**, **IN_MIN**, **IN_MAX**, **OUT_MIN** і **OUT_MAX**, виконати вимірювання значень змінних **_OUT_** і **_ERROR_**. Значення змінних **_OUT_** і **_ERROR_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.3.3. Перетворення діапазонів сигналу за допомогою функціонального блока перетворювач діапазонів сигналу LIN_TRAFO з використанням мови програмування Function Block Diagram (FBD)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram** (**FBD**).

2. Створити екземпляр **_LIN_TRAFO_** функціонального блока **LIN_TRAFO**.

3. Створити змінну **_OUT_** типу **REAL**.

4. Створити змінну **_ERROR_** типу **BOOL**.

5. Подати на вхід **IN:REAL** значення **0,5**.

6. Подати на вхід **IN_MIN:REAL** значення **0.0**.

7. Подати на вхід **IN_MAX:REAL** значення **1.0**.

8. Подати на вхід **OUT_MIN:REAL** значення **0.0**.

9. Подати на вхід **OUT_MAX:REAL** значення **10.0**.

10. Програмно зв’язати змінну **_OUT_** із виходом **OUT:REAL** екземпляра **_LIN_TRAFO_** функціонального блока **LIN_TRAFO** (див. рис. 23.2).

11. Програмно зв’язати змінну **_ERROR_** із виходом **ERROR:BOOL** екземпляра **_LIN_TRAFO_** функціонального блока **LIN_TRAFO** (див. рис. 23.2).

12. Запустити цей програмний код на виконання.

13. Змінюючи значення, які подаються на входи **IN**, **IN_MIN**, **IN_MAX**, **OUT_MIN** і **OUT_MAX**, виконати вимірювання значень змінних **_OUT_** і **_ERROR_**. Значення змінних **_OUT_** і **_ERROR_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

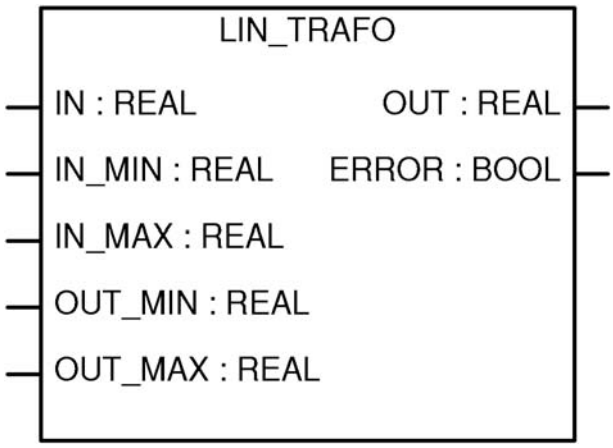



Рис. 23.3. Функціональний блок перетворювач діапазонів сигналу **LIN_TRAFO**

**2.4. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_INT (цілочисловий опис)**

**2.4.1. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_INT (цілочисловий опис) з використанням мови програмування Instruction List (IL)**

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

2. Створити екземпляр **_STATISTICS_INT_** функціонального блока **STATISTICS_INT**.

3. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

4. Створити змінну **_IN_** типу **ARRAY [0..9] OF INT** з початковими значеннями **–6**, **4**, **–4**, **8**, **0**, **2**, **6**, **–2**, **–8** і **10**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_MN_** типу **INT**.

7. Створити змінну **_MX_** типу **INT**.

8. Створити змінну **_AVG_** типу **INT**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_[_i_]
ST      _STATISTICS_INT_.IN
LD      _RESET_
ST      _STATISTICS_INT_.RESET
CAL     _STATISTICS_INT_
LD      _STATISTICS_INT_.MN
```

```
ST          _MN_
LD          _STATISTICS_INT_.MX
ST          _MX_
LD          _STATISTICS_INT_.AVG
ST          _AVG_
LD          _i_
ADD         1
MOD         10
ST          _i_
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значень змінних **_MN_**, **_MX_** і **_AVG_**. Значення змінних **_MN_**, **_MX_** і **_AVG_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.4.2. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_INT (цілочисловий опис) з використанням мови програмування Structured Text (ST)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_STATISTICS_INT_** функціонального блока **STATISTICS_INT**.

3. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

4. Створити змінну **_IN_** типу **ARRAY [0..9] OF INT** з початковими значеннями **–6**, **4**, **–4**, **8**, **0**, **2**, **6**, **–2**, **–8** і **10**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_MN_** типу **INT**.

7. Створити змінну **_MX_** типу **INT**.

8. Створити змінну **_AVG_** типу **INT**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_STATISTICS_INT_(IN:=_IN_[_i_], RESET:=_RESET_) ;
_MN_ := _STATISTICS_INT_.MN ;
_MX_ := _STATISTICS_INT_.MX ;
_AVG_ := _STATISTICS_INT_.AVG ;
_i_ := (_i_+1) MOD 10 ;
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значень змінних **_MN_**, **_MX_** і **_AVG_**. Значення змінних **_MN_**, **_MX_** і **_AVG_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.4.3. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_INT (цілочисловий опис) з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_STATISTICS_INT_** функціонального блока **STATISTICS_INT**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_IN_** типу **ARRAY [0..9] OF INT** з початковими значеннями **–6**, **4**, **–4**, **8**, **0**, **2**, **6**, **–2**, **–8** і **10**.

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

7. Створити змінну **_MN_** типу **INT**.

8. Створити змінну **_MX_** типу **INT**.

9. Створити змінну **_AVG_** типу **INT**.

10. Подати на вхід **IN:INT** значення **_IN_[_i_]**.

11. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_STATISTICS_INT_** функціонального блока **STATISTICS_INT** (див. рис. 23.4).

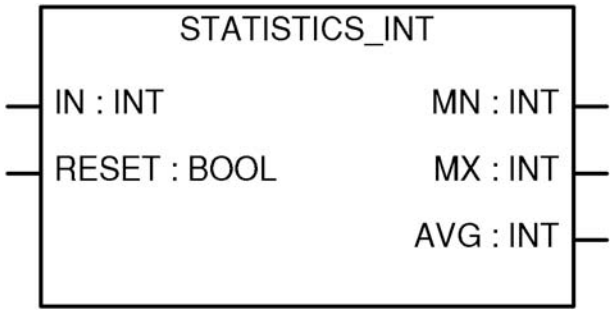



Рис. 23.4. Функціональний блок визначник значень сигналу **STATISTICS_INT** (цілочисловий опис)

12. Програмно зв’язати змінну **_MN_** із виходом **MN:INT** екземпляра **_STATISTICS_INT_** функціонального блока **STATISTICS_INT** (див. рис. 23.4).

13. Програмно зв’язати змінну **_MX_** із виходом **MX:INT** екземпляра **_STATISTICS_INT_** функціонального блока **STATISTICS_INT** (див. рис. 23.4).

14. Програмно зв’язати змінну **_AVG_** із виходом **AVG:INT** екземпляра **_STATISTICS_INT_** функціонального блока **STATISTICS_INT** (див. рис. 23.4).

15. Створити оператор **ADD**.

16. Програмно зв’язати змінну **_i_** із першим входом оператора **ADD**.

17. Подати на другий вхід оператора **ADD** значення **1**.

18. Створити оператор **MOD**.

19. Подати на перший вхід оператора **MOD** значення з виходу оператора **ADD**.

20. Подати на другий вхід оператора **MOD** значення **10**.

21. Програмно зв’язати змінну **_i_** із виходом оператора **MOD**.

22. Запустити цей програмний код на виконання.

23. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значень змінних **_MN_**, **_MX_** і **_AVG_**. Значення змінних **_MN_**, **_MX_** і **_AVG_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.5. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_REAL (дійсний опис)

#### 2.5.1. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_REAL (дійсний опис) з використанням мови програмування Instruction List (IL)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

2. Створити екземпляр **_STATISTICS_REAL_** функціонального блока **STATISTICS_REAL**.

3. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

4. Створити змінну **_IN_** типу **ARRAY [0..9] OF REAL** з початковими значеннями **–6.0**, **4.0**, **–4.0**, **8.0**, **0.0**, **2.0**, **6.0**, **–2.0**, **–8.0** і **10.0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_MN_** типу **REAL**.

7. Створити змінну **_MX_** типу **REAL**.

8. Створити змінну **_AVG_** типу **REAL**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_[_i_]
ST      _STATISTICS_REAL_.IN
LD      _RESET_
ST      _STATISTICS_REAL_.RESET
CAL     _STATISTICS_REAL_
LD      _STATISTICS_REAL_.MN
ST      _MN_
LD      _STATISTICS_REAL_.MX
ST      _MX_
LD      _STATISTICS_REAL_.AVG
ST      _AVG_
LD      _i_
ADD     1
MOD     10
```

**ST _i_**

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значень змінних **_MN_**, **_MX_** і **_AVG_**. Значення змінних **_MN_**, **_MX_** і **_AVG_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.5.2. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_REAL (дійсний опис) з використанням мови програмування Structured Text (ST)

1. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).

2. Створити екземпляр **_STATISTICS_REAL_** функціонального блока **STATISTICS_REAL**.

3. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

4. Створити змінну **_IN_** типу **ARRAY [0..9] OF REAL** з початковими значеннями **–6.0**, **4.0**, **–4.0**, **8.0**, **0.0**, **2.0**, **6.0**, **–2.0**, **–8.0** і **10.0**.

5. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_MN_** типу **REAL**.

7. Створити змінну **_MX_** типу **REAL**.

8. Створити змінну **_AVG_** типу **REAL**.

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_STATISTICS_REAL_(IN:=_IN_[_i_], RESET:=_RESET_) ;
_MN_ := _STATISTICS_REAL_.MN ;
_MX_ := _STATISTICS_REAL_.MX ;
_AVG_ := _STATISTICS_REAL_.AVG ;
_i_ := (_i_+1) MOD 10 ;
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значень змінних **_MN_**, **_MX_** і **_AVG_**. Значення змінних **_MN_**, **_MX_** і **_AVG_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.5.3. Визначення значень сигналу за допомогою функціонального блока визначник значень сигналу STATISTICS_REAL (дійсний опис) з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_STATISTICS_REAL_** функціонального блока **STATISTICS_REAL**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_IN_** типу **ARRAY [0..9] OF REAL** з початковими значеннями **–6.0**, **4.0**, **–4.0**, **8.0**, **0.0**, **2.0**, **6.0**, **–2.0**, **–8.0** і **10.0**.

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

7. Створити змінну **_MN_** типу **REAL**.

8. Створити змінну **_MX_** типу **REAL**.

9. Створити змінну **_AVG_** типу **REAL**.

10. Подати на вхід **IN:REAL** значення **_IN_[_i_]**.

11. Програмно зв'язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_STATISTICS_REAL_** функціонального блока **STATISTICS_RAL** (див. рис. 23.5).

12. Програмно зв'язати змінну **_MN_** із виходом **MN:REAL** екземпляра **_STATISTICS_REAL_** функціонального блока **STATISTICS_REAL** (див. рис. 23.5).

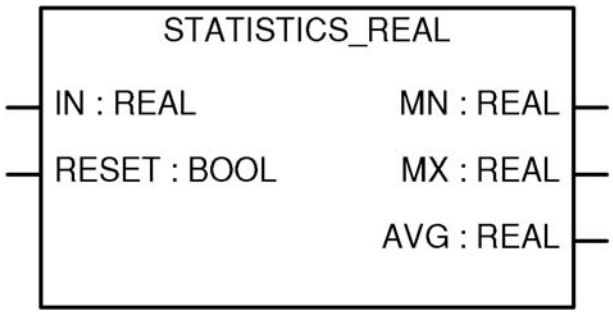



Рис. 23.5. Функціональний блок визначник значень сигналу **STATISTICS_REAL** (дійсний опис)

13. Програмно зв'язати змінну **_MX_** із виходом **MX:REAL** екземпляра **_STATISTICS_REAL_** функціонального блока **STATISTICS_REAL** (див. рис. 23.5).

14. Програмно зв'язати змінну **_AVG_** із виходом **AVG:REAL** екземпляра **_STATISTICS_REAL_** функціонального блока **STATISTICS_REAL** (див. рис. 23.5).

15. Створити оператор **ADD**.

16. Програмно зв'язати змінну **_i_** із першим входом оператора **ADD**.

17. Подати на другий вхід оператора **ADD** значення **1**.

18. Створити оператор **MOD**.

19. Подати на перший вхід оператора **MOD** значення з виходу оператора **ADD**.

20. Подати на другий вхід оператора **MOD** значення **10**.

21. Програмно зв'язати змінну **_i_** із виходом оператора **MOD**.

22. Запустити цей програмний код на виконання.

23. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значень змінних **_MN_**, **_MX_** і **_AVG_**. Значення змінних **_MN_**, **_MX_** і **_AVG_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.6. Визначення дисперсії сигналу за допомогою функціонального блока визначник дисперсії сигналу VARIANCE

#### 2.6.1. Визначення дисперсії сигналу за допомогою функціонального блока визначник дисперсії сигналу VARIANCE з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
3. Створити екземпляр **_VARIANCE_** функціонального блока **VARIANCE**.
4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.
5. Створити змінну **_IN_** типу **ARRAY [0..9] OF REAL** з початковими значеннями **–6.0**, **4.0**, **–4.0**, **8.0**, **0.0**, **2.0**, **6.0**, **–2.0**, **–8.0** і **10.0**.
6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
7. Створити змінну **_OUT_** типу **REAL**.
8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_[_i_]
ST      _VARIANCE_.IN
LD      _RESET_
ST      _VARIANCE_.RESET
CAL     _VARIANCE_
LD      _VARIANCE_.OUT
ST      _OUT_
LD      _i_
ADD     1
MOD     10
ST      _i_
```

9. Запустити цей програмний код на виконання.
10. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

#### 2.6.2. Визначення дисперсії сигналу за допомогою функціонального блока визначник дисперсії сигналу VARIANCE з використанням мови програмування Structured Text (ST)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text** (**ST**).
3. Створити екземпляр **_VARIANCE_** функціонального блока

**VARIANCE**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_IN_** типу **ARRAY [0..9] OF REAL** з початковими значеннями **–6.0**, **4.0**, **–4.0**, **8.0**, **0.0**, **2.0**, **6.0**, **–2.0**, **–8.0** і **10.0**.

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

7. Створити змінну **_OUT_** типу **REAL**.

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_VARIANCE_(IN:=_IN_[_i_], RESET:=_RESET_) ;
_OUT_ := _VARIANCE_.OUT ;
_i_ := (_i_+1) MOD 10 ;
```

9. Запустити цей програмний код на виконання.

10. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.6.3. Визначення дисперсії сигналу за допомогою функціонального блока визначник дисперсії сигналу VARIANCE з використанням мови програмування Function Block Diagram (FBD)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_VARIANCE_** функціонального блока **VARIANCE**.

4. Створити змінну **_i_** типу **DWORD** з початковим значенням **0**.

5. Створити змінну **_IN_** типу **ARRAY [0..9] OF REAL** з початковими значеннями **–6.0**, **4.0**, **–4.0**, **8.0**, **0.0**, **2.0**, **6.0**, **–2.0**, **–8.0** і **10.0**.

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

7. Створити змінну **_OUT_** типу **REAL**.

8. Подати на вхід **IN:REAL** значення **_IN_[_i_]**.

9. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_VARIANCE_** функціонального блока **VARIANCE** (див. рис. 23.6).

10. Програмно зв’язати змінну **_OUT_** із виходом **OUT:REAL** екземпляра **_VARIANCE_** функціонального блока **VARIANCE** (див. рис. 23.6).

11. Створити оператор **ADD**.

12. Програмно зв’язати змінну **_i_** із першим входом оператора **ADD**.

13. Подати на другий вхід оператора **ADD** значення **1**.

14. Створити оператор **MOD**.

15. Подати на перший вхід оператора **MOD** значення з виходу оператора **ADD**.

16. Подати на другий вхід оператора **MOD** значення **10**.

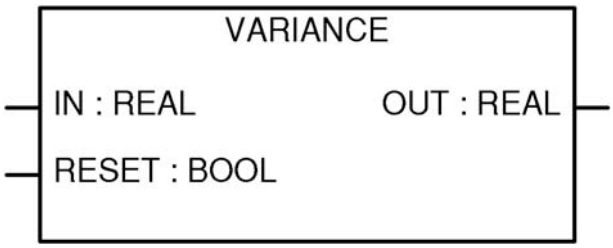



Рис. 23.6. Функціональний блок визначник дисперсії сигналу **VARIANCE**

17. Програмно зв’язати змінну **_i_** із виходом оператора **MOD**.

18. Запустити цей програмний код на виконання.

19. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на вхід **IN**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_** відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 3. Питання для самоперевірки

1. Які алгоритми використовуються для здійснення диференціювання сигналу?
2. Яка існує різниця між аналоговим (неперервним) і дискретним (цифровим) диференціюванням сигналів?
3. Як працює диференціатор сигналу **DERIVATIVE**?
4. Які алгоритми використовуються для здійснення інтегрування сигналу?
5. Яка існує різниця між аналоговим (неперервним) і дискретним (цифровим) інтегруванням сигналів?
6. Як працює інтегратор сигналу **INTEGRAL**?
7. Які алгоритми використовуються для здійснення перетворення діапазонів сигналу?
8. Як працює перетворювач діапазонів сигналу **LIN_TRAFO**?
9. Що таке математичне сподівання сигналу?
10. Що таке дисперсія сигналу?
11. Що таке середньоквадратичне сигналу?
12. Які алгоритми використовуються для здійснення статистичної обробки сигналів?
13. Як працює визначник значень сигналу **STATISTICS_INT** (цілочисловий опис)?
14. Як працює визначник значень сигналу **STATISTICS_REAL** (дійсний опис)?
15. Як працює визначник дисперсії сигналу **VARIANCE**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# ЛАБОРАТОРНА РОБОТА №24.
# ВИВЧЕННЯ РОБОТИ ГЕНЕРАТОРА BLINK, ЧАСТОТОМІРА FREQ_MEASURE І ГЕНЕРАТОРА GEN

## 1. Мета виконання лабораторної роботи

Метою виконання лабораторної роботи є вивчення роботи генератора **BLINK**, частотоміра **FREQ_MEASURE** і генератора **GEN**.

## 2. Порядок виконання лабораторної роботи

### 2.1. Формування сигналу прямокутної форми за допомогою функціонального блока генератор BLINK

#### 2.1.1. Формування сигналу прямокутної форми за допомогою функціонального блока генератор BLINK з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

3. Створити екземпляр **_BLINK_** функціонального блока **BLINK**.

4. Створити змінну **_ENABLE_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_OUT_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _ENABLE_
ST      _BLINK_.ENABLE
LD      T#2s
ST      _BLINK_.TIMELOW
LD      T#3s
ST      _BLINK_.TIMEHIGH
CAL     _BLINK_
LD      _BLINK_.OUT
ST      _OUT_
```

7. Запустити цей програмний код на виконання.

8. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **TIMELOW** і **TIMEHIGH**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.2. Формування сигналу прямокутної форми за допомогою функціонального блока генератор BLINK з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

3. Створити екземпляр **_BLINK_** функціонального блока **BLINK**.

4. Створити змінну **_ENABLE_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_OUT_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_BLINK_(ENABLE:=_ENABLE_,          TIMELOW:=T#2s,
TIMEHIGH:=T#3s) ;

_OUT_ := _BLINK_.OUT ;
```

7. Запустити цей програмний код на виконання.

8. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які подаються на входи **TIMELOW** і **TIMEHIGH**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.1.3. Формування сигналу прямокутної форми за допомогою функціонального блока генератор BLINK з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

3. Створити екземпляр **_BLINK_** функціонального блока **BLINK**.

4. Створити змінну **_ENABLE_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

5. Створити змінну **_OUT_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

6. Програмно зв’язати змінну **_ENABLE_** із входом **ENABLE:BOOL** екземпляра **_BLINK_** функціонального блока **BLINK** (див. рис. 24.1).

7. Подати на вхід **TIMELOW:TIME** значення **T#2s**.

8. Подати на вхід **TIMEHIGH:TIME** значення **T#3s**.

9. Програмно зв’язати змінну **_OUT_** із виходом **OUT:BOOL** екземпляра **_BLINK_** функціонального блока **BLINK** (див. рис. 24.1).

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічного або електронного контактного перетворювача, підключеного до дискретного входу **DI–1–0**, а також значення, які по-

даються на входи **TIMELOW** і **TIMEHIGH**, виконати вимірювання стану дискретного виходу **DO–1–0** (електромагнітного реле). Стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

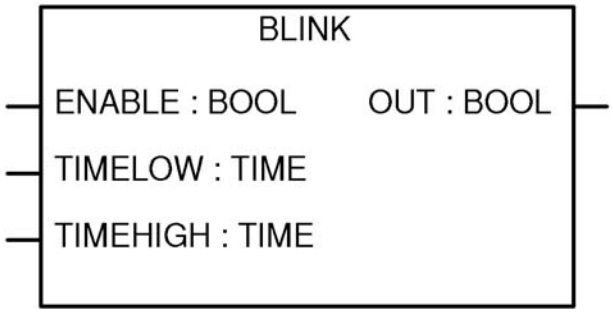



Рис. 24.1. Функціональний блок генератор **BLINK**

**2.2. Вимірювання частоти сигналу за допомогою частотоміра FREQ_MEASURE**

**2.2.1. Вимірювання частоти сигналу за допомогою частотоміра FREQ_MEASURE з використанням мови програмування Instruction List (IL)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).

4. Створити екземпляр **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE**.

5. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_OUT_** типу **REAL**.

8. Створити змінну **_VALID_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      _IN_
ST      _FREQ_MEASURE_.IN
LD      5
ST      _FREQ_MEASURE_.PERIODS
LD      _RESET_
ST      _FREQ_MEASURE_.RESET
CAL     _FREQ_MEASURE_
```

**LD** **_FREQ_MEASURE_.OUT**

**ST** **_OUT_**

**LD** **_FREQ_MEASURE_.VALID**

**ST** **_VALID_**

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PERIODS**, виконати вимірювання значення змінної **_OUT_**, а також стану дискретного виходу **DO–1–0 (**електромагнітного реле). Значення змінної **_OUT_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.2.2. Вимірювання частоти сигналу за допомогою частотоміра FREQ_MEASURE з використанням мови програмування Structured Text (ST)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

4. Створити екземпляр **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE**.

5. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_OUT_** типу **REAL**.

8. Створити змінну **_VALID_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

9. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_FREQ_MEASURE_(IN:=_IN_,                PERIODS:=5,
RESET:=_RESET_) ;
_OUT_ := _FREQ_MEASURE_.OUT ;
_VALID_ := _FREQ_MEASURE_.VALID ;
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PERIODS**, виконати вимірювання значення змінної **_OUT_**, а також стану дискретного виходу **DO–1–0 (**електромагнітного реле). Значення змінної **_OUT_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

**2.2.3. Вимірювання частоти сигналу за допомогою частотоміра FREQ_MEASURE з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

4. Створити екземпляр **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE**.

5. Створити змінну **_IN_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_OUT_** типу **REAL**.

8. Створити змінну **_VALID_** типу **BOOL** із адресою **%QX1.0** (яка відповідає дискретному виходу **DO–1–0**).

9. Програмно зв’язати змінну **_IN_** із входом **IN:BOOL** екземпляра **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE** (див. рис. 24.2).

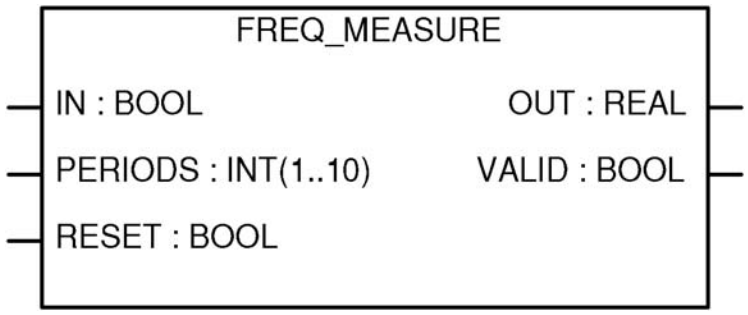



Рис. 24.2. Функціональний блок частотомір **FREQ_MEASURE**

10. Подати на вхід **PERIODS:INT(1..10)** значення **5**.

11. Програмно зв’язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE** (див. рис. 24.2).

12. Програмно зв’язати змінну **_OUT_** із виходом **OUT:BOOL** екземпляра **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE** (див. рис. 24.2).

13. Програмно зв’язати змінну **_VALID_** із виходом **VALID:BOOL** екземпляра **_FREQ_MEASURE_** функціонального блока **FREQ_MEASURE** (див. рис. 24.2).

14. Запустити цей програмний код на виконання.

15. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, яке подається на вхід **PERIODS**, виконати вимірювання значення

змінної **_OUT_**, а також стану дискретного виходу **DO–1–0 (**електромагнітного реле). Значення змінної **_OUT_**, а також стан дискретного виходу **DO–1–0** (електромагнітного реле) відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію).

### 2.3. Формування сигналу стандартних форм за допомогою генератора GEN

#### 2.3.1. Формування сигналу стандартних форм за допомогою генератора GEN з використанням мови програмування Instruction List (IL)

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.
2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.
3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Instruction List** (**IL**).
4. Створити екземпляр **_GEN_** функціонального блока **GEN**.
5. Створити змінну **_BASE_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).
6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).
7. Створити змінну **_OUT_** типу **INT**.
8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
LD      TRIANGLE
ST      _GEN_.MODE
LD      _BASE_
ST      _GEN_.BASE
LD      T#5s
ST      _GEN_.PERIOD
LD      500
ST      _GEN_.CYCLES
LD      10
ST      _GEN_.AMPLITUDE
LD      _RESET_
ST      _GEN_.RESET
CAL     _GEN_
LD      _GEN_.OUT
ST      _OUT_
```

9. Запустити цей програмний код на виконання.
10. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **MODE**, **PERIOD**, **CYCLES** і **AMPLITUDE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_**відобразити як в текстовому вигляді, так і в графічному вигляді (вико-

ристовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.3.2. Формування сигналу стандартних форм за допомогою генератора GEN з використанням мови програмування Structured Text (ST)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Structured Text (ST)**.

4. Створити екземпляр **_GEN_** функціонального блока **GEN**.

5. Створити змінну **_BASE_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_OUT_** типу **INT**.

8. Увести програмний код, наведений нижче, в програму **PLC_PRG**.

```
_GEN_(MODE:=TRIANGLE, BASE:=_BASE_, PERIOD:=T#5s,
CYCLES:=500, AMPLITUDE:=10, RESET:=_RESET_) ;
_OUT_ := _GEN_.OUT ;
```

10. Запустити цей програмний код на виконання.

11. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **MODE**, **PERIOD**, **CYCLES** і **AMPLITUDE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_**відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

**2.3.3. Формування сигналу стандартних форм за допомогою генератора GEN з використанням мови програмування Function Block Diagram (FBD)**

1. Підключити до дискретного входу **DI–1–0** механічний або електронний контактний перетворювач.

2. Підключити до дискретного входу **DI–2–0** механічний або електронний контактний перетворювач.

3. Створити проект, першим **POU** якого є програма з ідентифікатором **PLC_PRG** на мові програмування **Function Block Diagram (FBD)**.

4. Створити екземпляр **_GEN_** функціонального блока **GEN**.

5. Створити змінну **_BASE_** типу **BOOL** із адресою **%IX0.0** (яка відповідає дискретному входу **DI–1–0**).

6. Створити змінну **_RESET_** типу **BOOL** із адресою **%IX0.1** (яка відповідає дискретному входу **DI–2–0**).

7. Створити змінну **_OUT_** типу **INT**.

8. Подати на вхід **MODE:GEN_MODE** значення **TRIANGLE**.

9. Програмно зв’язати змінну **_BASE_** із входом **BASE:BOOL** екземпляра **_GEN_** функціонального блока **GEN** (див. рис. 24.3).

10. Подати на вхід **PERIOD:TIME** значення **T#5s**.
11. Подати на вхід **CYCLES:INT** значення **500**.
12. Подати на вхід **AMPLITUDE:INT** значення **10**.
13. Програмно зв'язати змінну **_RESET_** із входом **RESET:BOOL** екземпляра **_GEN_** функціонального блока **GEN** (див. рис. 24.3).
14. Програмно зв'язати змінну **_OUT_** із виходом **OUT:INT** екземпляра **_GEN_** функціонального блока **GEN** (див. рис. 24.3).
15. Запустити цей програмний код на виконання.

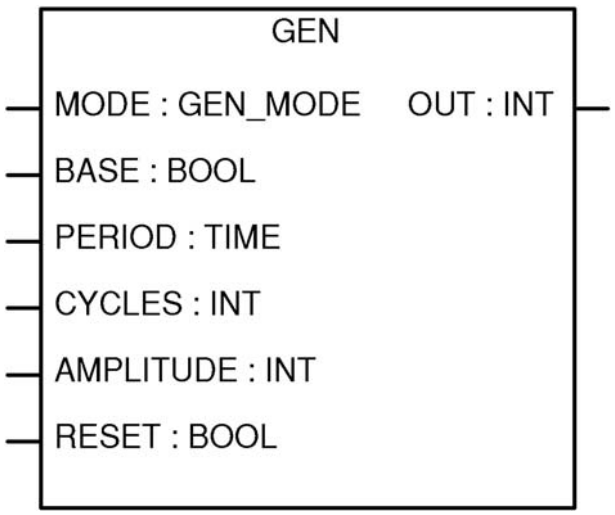



Рис. 24.3. Функціональний блок генератор **GEN**

16. Змінюючи стан механічних або електронних контактних перетворювачів, підключених відповідно до дискретних входів **DI–1–0** і **DI–2–0**, а також значення, які подаються на входи **MODE**, **PERIOD**, **CYCLES** і **AMPLITUDE**, виконати вимірювання значення змінної **_OUT_**. Значення змінної **_OUT_**відобразити як в текстовому вигляді, так і в графічному вигляді (використовуючи візуалізацію і цифрове трасування **Sampling Trace**).

### 3. Питання для самоперевірки

1. Яку форму і параметри має прямокутний випробувальний сигнал?
2. Як працює генератор **BLINK**?
3. Які існують методи вимірювання частоти сигналу?
4. Як працює частотомір **FREQ_MEASURE**?
5. Яку форму і параметри має синусоїдний випробувальний сигнал?
6. Яку форму і параметри має трикутний випробувальний сигнал?
7. Яку форму і параметри має пилкоподібний випробувальний сигнал?
8. Як працює генератор **GEN**?

### 4. Рекомендована література

1. Основна література [1о–8о].
2. Додаткова література [1д–3д].

# СПИСОК ЛІТЕРАТУРИ

## Основна література


1. Васильківський І.С. Виконавчі пристрої систем автоматизації. Навчальний посібник / І.С. Васильківський, В.О. Фединець, Я.П. Юсик. – Львів: Львівська політехніка, 2020. – 220 с. – ISBN 978–966–941–543–1.
2. Грицунов О.В. Інформаційні системи та технології. Навчальний посібник. – Х.: ХНАМГ, 2010. – 222 с. – ISBN 978–966–695–195–6.
3. Ельперін І.В. Промислові контролери: Навчальний посібник / І.В. Ельперін – К.: НУХТ, 2003. – 320 с. – ISBN 966–612–024–0.
4. Ладанюк А.П. Автоматизація технологічних процесів і виробництв харчової промисловості: Підручник / А.П. Ладанюк, В.Г. Трегуб, І.В. Ельперін, В.Д. Цюцюра. – К.: Аграрна освіта, 2001. – 224 с. – ISBN 966–95661–2–6.
5. Лисаченко І. Г. Програмне забезпечення комп'ютерно-інтегрованих систем управління хіміко-технологічними процесами: Навчально-методичний посібник / І.Г. Лисаченко. – Х.: НТУ "ХПІ", 2012. – 112 с. – ISBN 000-000-000-000-0.
6. Пушкар М.С. Проектування систем автоматизації: Навч. посібник / М.С. Пушкар, С.М. Проценко. – Д.: Національний гірничий університет, 2013. – 268 с. – ISBN 978–966–350–423–0.
7. Савицький В. Технічні засоби автоматизації / В. Савицький, Р. Федоришин. – Львів: Львівська політехніка, 2018. – 292 с. – ISBN 978–966–941–182–2.
8. Трегуб В.Г. Проектування систем автоматизації: Навч. посібник. – К.: Видавництво Ліра-К, 2017. – 344 с. – ISBN 978–966–2609–58–5.


## Додаткова література


1. ОВЕН ПЛК150. Контролер програмований логічний. Керівництво з експлуатації АРАВ.421445.002 КЕ.
2. ГОСТ 26.011-80 Засоби вимірювань і автоматизації. Сигнали струму і напруги електричні неперервні вхідні і вихідні.
3. Конфігурування області уведення/виведення ОВЕН ПЛК100/ПЛК150/ ПЛК154. Керівництво користувача.