

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

**РОЗРОБЛЕННЯ ППЗ
ДЛЯ ПРОМИСЛОВИХ КОНТРОЛЕРІВ ОВЕН
В СЕРЕДОВИЩІ CODESYS V2.3**

Методичні вказівки
для проведення комп'ютерного практикуму з курсу
«Програмне забезпечення промислових контролерів»

для студентів спеціальності 151
«Автоматизація та комп'ютерно-інтегровані технології»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 16.01.2019 р.

Харків
НТУ «ХП»
2019

Розроблення ППЗ для промислових контролерів ОВЕН в середовищі *CODESYS V2.3*: Методичні вказівки для проведення комп'ютерного практикуму з курсу «Програмне забезпечення промислових контролерів» для студентів спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання / Уклад.: Подустов М.О., Бабіченко А.К., Лисаченко І. Г., Дзевочко А.І. – Харків.: НТУ «ХП», 2019. – 80 с.

Укладачі: М.О. Подустов
А. К. Бабіченко
І. Г. Лисаченко
А. І. Дзевочко

Рецензент: І. Л. Красніков, доц. канд. техн. наук,
проф. каф. автоматизації технологічних систем
та екологічного моніторингу

Кафедра автоматизації технологічних систем
та екологічного моніторингу

ВСТУП

Методичні вказівки розроблені для проведення комп'ютерного практикуму з дисципліни «Програмне забезпечення промислових контролерів» зі студентами спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології». Вони вміщують необхідні теоретичні відомості для розробки прикладного програмного забезпечення (ППЗ) для промислових контролерів.

Для розроблення та завантаження ППЗ до складу програмно-технічного комплексу (ПТК) входить ПК зі встановленим спеціальним програмним забезпеченням (СПЗ) – комплексом *CoDeSys V2.3* [1, 2] (скорочення від *Controllers Development System*), який розроблений компанією *3S - Smart Software Solutions GmbH* (Німеччина) [3]. В подальшому в тексті будемо використовувати скорочену назву середовища – *CDS2*. Апаратною складовою ПТК є промислові контролери ОВЕН ПЛК 100-ї серії виробництва компанії ТОВ «ВО ОВЕН» (Україна) [4]. Опис основних принципів роботи з контролерами ОВЕН наведено в роботах [5, 6], а технічну підтримку можна одержати за адресою в *Internet* [4].

Комп'ютерний практикум проводиться на стендах, які мають у своєму складі програмованій логічний контролер (ПЛК), об'єкт управління (модель теплообмінника) та дискретні і аналогові датчики. Модель теплообмінника складається з нагрівача (нагрівальний резистор) та охолоджувача (вентилятор обдування). Контрольованим параметром є температура повітря всередині теплообмінника, яка вимірюється за допомогою або термометра опору, або термоелектричного перетворювача – термопары. Також на стенді є імітатор завдання для контрольованого параметру або датчик положення засувки – змінний резистор. Для імітування сигнала

лів управління та дискретних датчиків до ПЛК підключений емулятор дискретних сигналів з перемикачами типу «сухий контакт».

Метою проведення комп'ютерного практикуму є закріплення теоретичних знань та отримання практичних навичок програмування ПЛК під час розроблення систем управління технологічними процесами. Завдання комп'ютерного практикуму розроблені за принципом поступового їхнього ускладнення: від звичайного логічного керування до реалізації законів регулювання (ПД- або двопозиційного закону регулювання) технологічного параметра, які здійснює ПЛК. Виконання комп'ютерного практикуму забезпечує засвоєння метода аналогового керування дискретним вихідним елементом, так зване ШІМ-регулювання. Також в методичних вказівках розглянуті питання розроблення візуалізації стану технологічного обладнання, тобто ППЗ для людино-машинного інтерфейсу оператора автоматизованого робочого місця (АРМ) технологічної установки, яке сумісно з промисловим контролером є складовими елементами розподілених систем управління (РСУ).

Практичне завдання 1
ЗАГАЛЬНІ ПРИНЦИПИ ПРОГРАМУВАННЯ
КОНТРОЛЕРІВ ОВЕН В СЕРЕДОВИЩІ CODESYS V2.3

1.1. Мета завдання

– ознайомлення з обладнанням стенда, вивчення його принципової та структурної схем;

– закріплення теоретичних знань щодо побудови програмованих логічних контролерів *ОВЕН* серій *ПЛК100/150/154* та структури програмного середовища *CDS2*;

– отримання практичних навичок та прийомів роботи в середовищі *CDS2* в процесі створення ППЗ для ПЛК: *створення проекту, конфігурування ресурсів контролера, написання програми користувача, компіляція та емуляція роботи програми користувача, підключення до ПЛК та завантаження проекту до нього.*

1.2. Опис стенда

Загальний вигляд стенда зображено на рис. 1.1. Насамперед відмітимо, що стенд розроблений для виконання комп'ютерних практикумів з багатьох дисциплін. Тому до його складу входять елементи, які не використовують під час вивчення дисципліни «Програмне забезпечення промислових контролерів».

На рис. 1.1 виносками позначені: *1* – імітатор нагрівача, *2* – імітатор охолоджувача, *3* – клавіша вмикання живлення операторської панелі (ОП), *4* – імітатор аналогового сигналу опору (*R~*), *5* – ПЛК, *6* та *7* – імітатори дискретних сигналів, *8* – ОП, *9* – клавіша вмикання живлення ПЛК, *10* та *11* – датчики температури. Нижче наведено короткий опис елементів стенда:

1. Імітатор нагрівача – потужний дрововий опір в керамічному корпусі типу *ПЭВ-100* номіналом *750 Ом* та живленням *АС220V*.

2. Імітатор охолоджувача – вентилятор обдування від звичайного ПК з живленням *DC12V*.

3, 9. Клавіші вмикання живлення ОП та ПЛК.

4. Імітатор аналогового сигналу – змінний опір номіналом *0...1 кОм*.

5. Контролер *ОВЕН* моделі *ПЛК150-220.И-L* з внутрішнім джерелом живлення, який є моноблоком, що об'єднує в собі контактні групи для підключення дискретних та аналогових сигналів введення/виведення, а також інтерфейси обміну: *RS-232* – для завантаження програм та обміну даними з ПК; *RS-485* – для мережного обміну з іншими приладами, наприклад, з ОП; *Ethernet* – для завантаження програм та обміну даними з ПК та іншими ПЛК.

6. Імітатор вхідних дискретних сигналів *ЭДИ-6* для ПЛК – шість перемикачів типу «сухий контакт».

7. Імітатор вхідних дискретних сигналів для ОП – чотири перемикачі типу «сухий контакт» (в практикумі не задіяні).

8. Панель оператора *ОВЕН СМІ-1-220* для індикації даних з функціями редагування і для роботи в мережах *RS-485* та *RS-232* за протоколами *ModBus ASCII/RTU* та *OWEN* (в практикумі не задіяна).

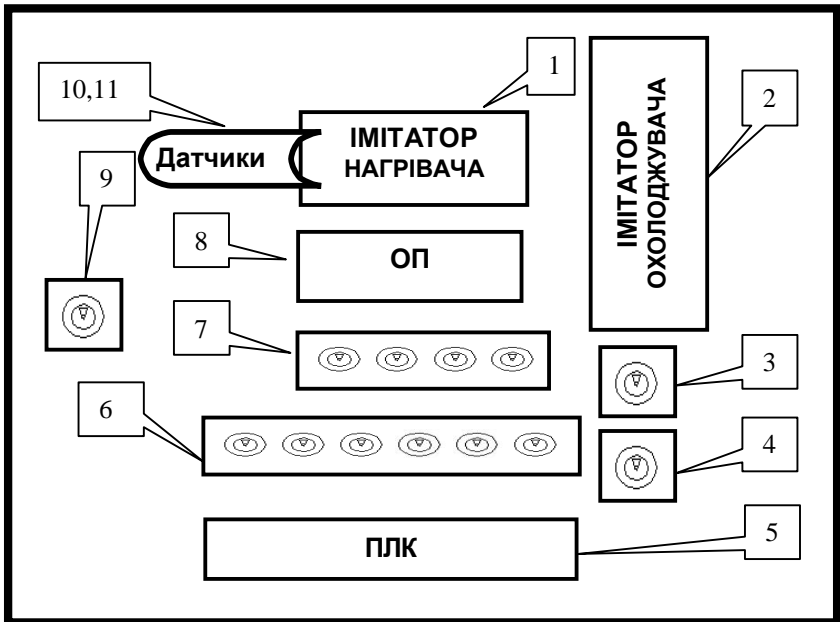


Рис. 1.1 – Загальний вигляд стенда

10, 11. Датчики температури – термоперетворювач опору з НСХ $Pt100$ ($k=0,00385^{\circ}\text{C}^{-1}$) та термоелектричний перетворювач (термопара) типу K (ТХА – термопара з робочим спаям хромель-алюмель).

На рис. 1.2 зображено принципову електричну схему стенда. Структурна схема комунікаційних зв'язків елементів стенда (ПЛК та ОП) з ПК зображена на рис. 1.3.

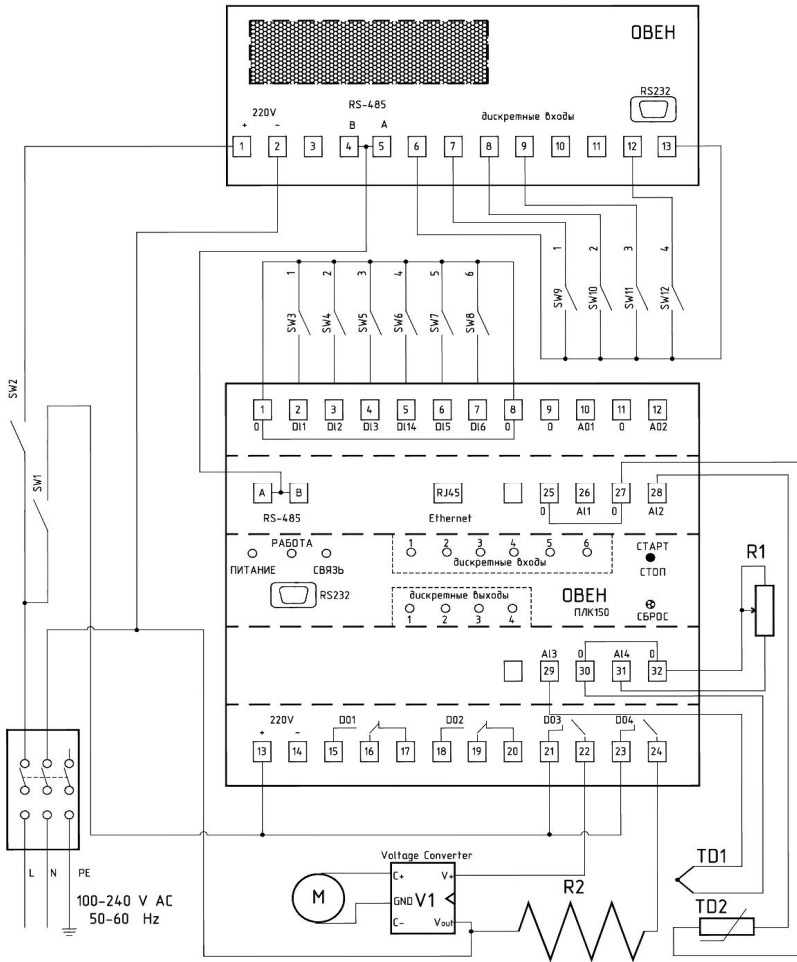


Рис. 1.2 – Принципова електрична схема стенда

Згідно з рис. 1.2 принцип роботи стенда є таким. Перемикачі *SW7*, *SW8* призначені для вмикання живлення ПЛК та ОП. За допомогою перемикачів *SW1...SW6* дискретні сигнали подаються на входи *DI1...6* ПЛК, а за допомогою перемикачів *SW9...SW12* – на входи *Bx.1*, *Bx.2*, *Bx.3* та *Bx.6* ОП. Ці перемикачі можуть імітувати сигнали дискретних датчиків («ВМК.»/«ВИМК.»), управляти режимами роботи («Руч.»/«Автомат.») або технологічним процесом («Більше»/«Менше»). Імітатори нагрівача та охолоджувача підключені до дискретних виходів контролера, причому нагрівач підключений безпосередньо до релейного виходу *DO4* ПЛК, а охолоджувач, через перетворювач напруги *AC220V/DC12V* – до виходу *DO3* ПЛК. Змінний опір підключений до четвертого аналогового входу ПЛК *AI4*. До другого аналогового входу ПЛК *AI2* підключений термоперетворювач опору *Pt100* (TD2 на рис. 1.2). Термопара типу «К» підключена до третього аналогового входу ПЛК *AI3* (TD1 на рис. 1.2).

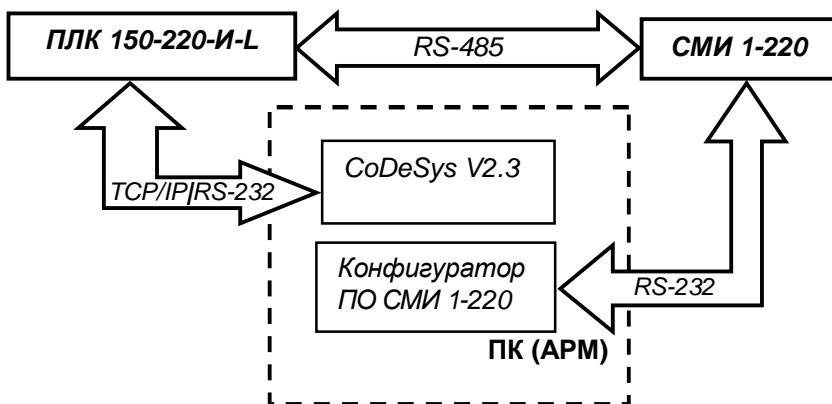


Рис. 1.3 – Структурна схема стенда

Нагадаємо, що при проведенні практикуму операторська панель та дискретні перемикачі, які до неї приєднані, не використовують.

1.3. Порядок виконання завдання

Виконання практичного завдання складається з таких етапів:

- 1) Створення проекту в *CDS2*:

- вибір цільового ПЛК (*target*-файла) та мови програмування головного програмного організаційного компоненту (*POU*) *PLC_PRG*, збереження проекту на жорсткому диску ПК;
- конфігурування ПЛК у вкладці *Resources* проекту;
- підключення додаткових бібліотек (за необхідністю);
- розроблення програми користувача;
- компілювання проекту та перевірка відповідності роботи алгоритму до завдання на симуляторі;
- створення візуалізації для налагодження проекту.

2) Підключення середовища програмування *CDS2* (ПК) до ПЛК та завантаження до нього машинного коду проекту.

3) Налагодження проекту в покроковому режимі та перевірка роботи програми користувача для управління технологічним процесом.

1.4. Хід виконання завдання

1.4.1. Створення проекту в середовищі CoDeSys V2.3

Після встановлення середовища *CDS2* на ПК необхідно до нього підключити потрібні *target*-файли (тобто опис структури та ресурсів ПЛК, які надаються виробником). Ця процедура здійснюється за допомогою програми *InstallTarget.exe*, яка знаходиться в тій директорії, що й файл для запуску середовища *CDS2*. Далі потрібно діяти таким чином. По-перше, за допомогою кнопки ... в верхній частині вікна вибрати директорію, де зберігаються раніше скопійовані *target*-файли та натиснути на кнопку ОК. По-друге, використовуючи кнопку *Open...*, знайти каталог з потрібним *target*-файлом. В результаті, ліворуч у вікні з ім'ям *Possible Targets* з'явиться потрібна директорія (*PLC150.l_L*) з файлом *plc.trg*. По-третє, натиснути лівою кнопкою миші (ЛКМ) на кнопку *Install* і далі – на кнопку *Close*. На рис. 1.4 зображено вікно програми *InstallTarget* з потрібним *target*-файлом перед його завантаженням.

Отже, зараз все готово для подальшого виконання завдання практикуму. Запустити на виконання програмне середовище *CDS2*. Для цього використайте ярлик на робочому столі ПК або запустіть файл *Codesys.exe* в каталозі зі шляхом доступу *C:\Program Files\3S Software\CoDeSys V2.3*.**. Також *CDS2* можна запустити за допомогою програмної кнопки *ПУСК* та послідовного вибору шляху до потрібного файлу запуску середовища *CDS2*.

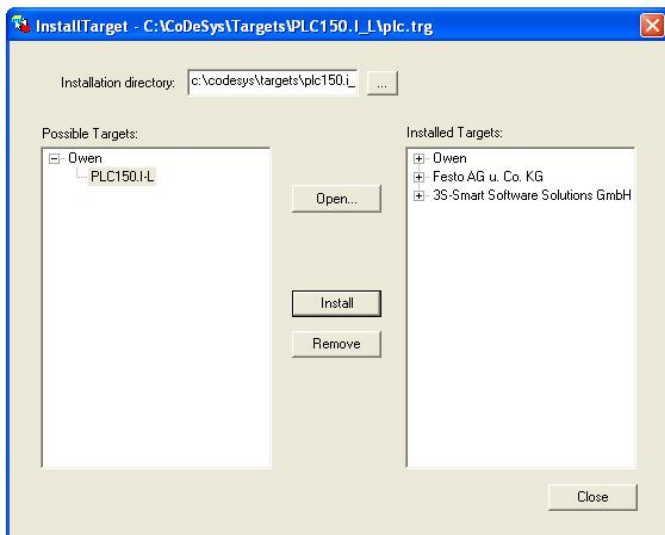


Рис. 1.4 – Вікно з потрібним *target*-файлом перед його встановленням в середовище *CoDeSys V2.3*

Якщо в загальних властивостях середовища *CDS2* не активована функція завантаження останнього проекту або у випадку першого запуску після встановлення середовища, то відкриється пусте вікно. Зовнішній вигляд вперше запущеного середовища *CDS2* зображений на рис. 1.5.

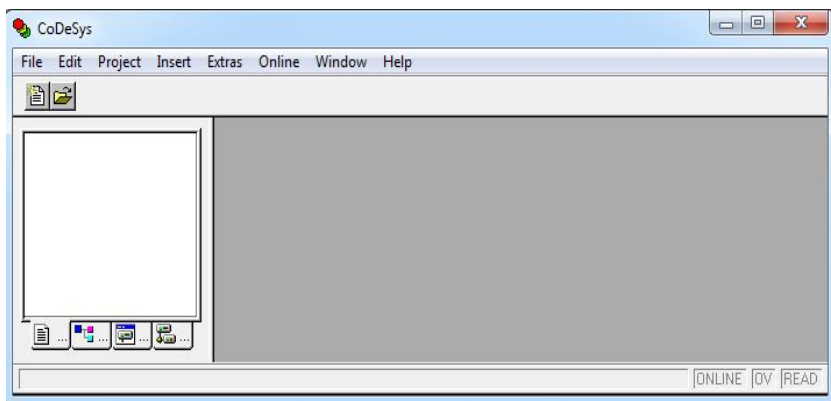


Рис. 1.5 – Вікно вперше запущеного середовища *CoDeSys V2.3*

Почніть розроблення проекту в середовищі *CDS2*. В меню **File** виберіть команду **New** або натисніть ЛКМ на відповідну кнопку в панелі інструментів. В наступному вікні **Target Settings** за допомогою кнопки «▼» праворуч від рядка **Configuration**, виберіть в випадаючому вікні потрібну модель контролера (в нашому випадку це буде *PLC 150-1.L*). Відкриється вікно для налаштувань параметрів цільового ПЛК вибраної моделі. Залишіть ці параметри без змін. Підтвердить свій вибір цільового контролера натисненням ЛКМ на кнопку **OK** праворуч.

Далі, у вікні **New POU** виберіть тип *POU* – головну програму з ім'ям **PLC_PRG**, а також мову реалізації *CFC* (графічна мова, яка використовує вільно розташовані блочні елементи та зв'язки між ними). Зауважимо, що ім'я головної програми змінювати заборонено для проектів з одним головним завданням для виконання програми користувача. Після всіх проведених операцій відкриється робоче вікно середовища програмування. Зовнішній вигляд вікна програмного середовища з активним редактором *POU* з ім'ям **PLC_PRG** зображено на рис. 1.6.

Використовуючи в меню **File** стандартні команди **Save**, або **Save as...**, або відповідну кнопку на панелі інструментів, збережіть проект на жорсткому диску ПК. Надайте ім'я проекту, наприклад, таке: **lg_1_name.pro** (замість **name** вкажіть своє прізвище).

1.4.2 Конфігурування ресурсів ПЛК

Наступним кроком в процесі розроблення проекту є конфігурування ресурсів фіксованого набору модулів ПЛК (налаштування дискретних та аналогових входів/виходів). Всі змінні, які оголошені в ресурсах ПЛК, автоматично є глобальними, причому обмін даними з програмою користувача з боку системи виконання здійснюється через спеціальну область пам'яті контролера – *пам'ять введення/виведення*.

ВАЖЛИВО. Розмір пам'яті введення/виведення може бути обмежений ліцензією *CoDeSys*, що встановлена на контролер *ПЛК1xx*. Ліцензійне обмеження розповсюджується на контролери, в назві моделі яких останньою буквою стоїть латинська літера «L». Тоді пам'ять введення/виведення обмежена розміром 360 байтів. При цьому 122 байти відводяться для пам'яті введення (%I), 234 байти – під пам'ять

виведення (%Q), а 4 байти, що залишилися – під спеціальну меркерну пам'ять (%M). Для контролерів з останньою літерою в назві яких стоїть латинська «М» обмежень у розмірі пам'яті введення/виведення немає. За умовчанням сумарний обсяг пам'яті введення (%I) та виведення (%Q) встановлений таким, що дорівнює 16 кБ, а це достатньо для більшості завдань [5].

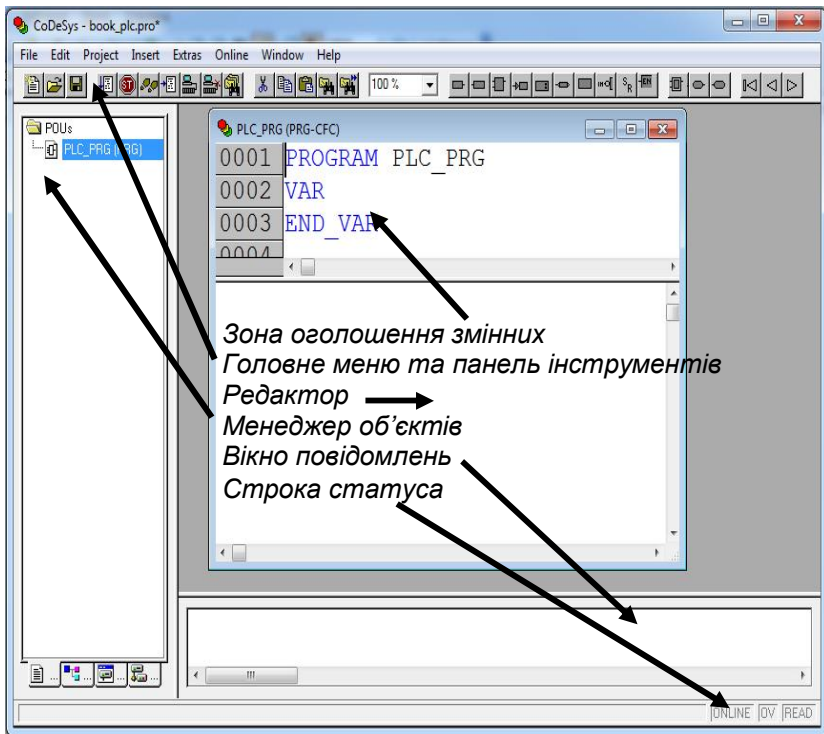


Рис. 1.6 – Вікно з відкритим проектом

Отже, в середовищі *CDS2* був створений проект з ім'ям *lr_1_name.pro* для цільової платформи *ОВЕН150-І.І* та мовою програмування *CFC*. Перейдіть до вкладення *Resources* та виберіть утиліту *PLC Configuration*. Взагалі для конфігурування ресурсів ПЛК надається докладна інструкція [6] від виробника контролерів *ОВЕН*.

Спочатку розглянемо деякі терміни, структуру дерева конфігурації та пояснімо дії в процесі конфігурування ресурсів ПЛК.

Конфігурація ПЛК – це опис структури та складу апаратних ресурсів ПЛК. Вона зчитується середовищем *CDS2* з *target*-файла відповідного ПЛК.

Конфігурування ПЛК – це процес налаштування ресурсів та вибору параметрів ПЛК.

Дерево конфігурації ПЛК – це результат зчитування та декодування середовищем *CDS2 target*-файла відповідного ПЛК. Дерево конфігурації складається з фіксованих та змінюваних модулів. Фіксовані модулі складаються з окремих «каналів». Якщо модуль має декілька каналів, то його називають «слот». Однакові модулі мають однаковий склад, але прив'язані к власним фізичним каналам введення/виведення та мають власні зони пам'яті ПЛК з абсолютною адресою.

Праворуч від дерева розміщені вкладення **Settings** та **Module parameters**. Вкладення **Settings** має фіксовані значення параметрів, котрі неможливо змінити. Але можливе встановлення показників для активування автоматичного розподілу адрес, перевірки перетину адрес та функції збереження конфігурації в проекті. Вкладення **Module parameters** має перелік категорій, які можуть змінюватись. Категорії розміщені рядками. Стовпчики відповідної категорії відображають номер категорії, її назву та значення. Лише в третьому стовпчику **Value** можливе встановлення значення параметра відповідної категорії. В четвертому стовпчику зазначено значення параметру за умовчанням. Останні п'ятий та шостий стовпчики вказують на можливі мінімальне та максимальне значення параметру. Насамкінець, кожен канал або слот конфігурації ПЛК має свій власний перелік параметрів.

Розглянемо порядок конфігурування ресурсів ПЛК. Отже, для конфігурування ресурсів ПЛК відкрийте вікно **PLC Configuration**. Для розгортання дерева конфігурації ПЛК потрібно натиснути ЛКМ на символ «+». З'явиться дерево з фіксованим набором ресурсів ПЛК та вкладеннями **Settings** та **Module parameters**, екранна форма якого зображена на рис. 1.7.

По-перше, встановіть мінімальний час циклу ПЛК. Спочатку ЛКМ позначте в дереві конфігурації ПЛК головний модуль – PLC 150 I. Далі, у вкладенні **Module parameters** ЛКМ клікніть по місцю перетину рядка **MinCycleLength** та стовпчика **Value**, а в рамці з пунктиру, що з'явиться, встановіть значення часу циклу **10 ms** замість значення за умовчанням **1 ms**. Якщо встановити цей параметр рівним нулю, тоді ПЛК буде виконувати програму користувача з циклом, який не є фіксованим. Це означає, що цикл сканування буде починатись відразу після закінчення попереднього циклу.

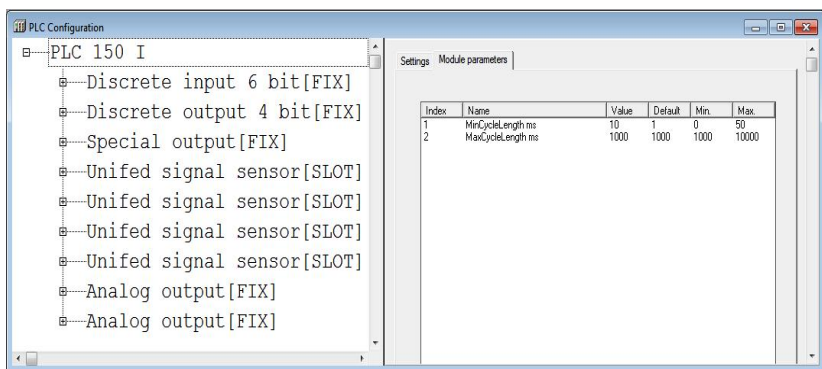


Рис. 1.7 – Вікно з деревом ресурсів ПЛК

По-друге, проведіть налаштування каналів дискретних входів/виходів та аналогових входів контролера. Для цього, подібно до попередніх дій натисніть ЛКМ на символи «+». Відкриються слоти *Discrete input 6 bit (FIX)* – з каналами дискретних входів та *Discrete output 6 bit (FIX)* – з каналами дискретних виходів. Привласнення символного імені каналу (входу або виходу) проводиться так: подвійним кліком ЛКМ натисніть на літери «АТ» у потрібному рядку каналу. Відкриється порожнє поле, яке по контуру позначено точками. До цього поля введіть символне ім'я змінної. В імені змінної можливе використання латинських літер, роздільника (нижній штрих), деяких символів та цифр. Причому починати символне ім'я з цифри заборонено.

Нехай для управління установкою буде використаний перемикач

типу «сухий контакт», який підключений до одного з дискретних входів ПЛК за допомогою емулятора сигналів *ЭДИ-6*. Надайте першому дискретному входу символічне ім'я **DIN1**. Подібно до попереднього призначення надайте символічні імена п'яти входам, що залишились (**DIN2**, **DIN3**, **DIN4**, **DIN5** та **DIN6**). Далі налаштуйте чотири вихідні канали з підключеними до них вихідними пристроями, якими буде управляти ПЛК згідно з програмою користувача. Для цього привласніть символічні імена вихідним каналам, наприклад **DOUT1**, **DOUT2**, **DOUT3** та **DOUT4**. В результаті буде отримана конфігурація дискретних вхідних та вихідних каналів (див. рис. 1.8). Відмітимо, що регістр літер в імені змінної не має значення. В рядку поруч з символічною адресою для зберігання стану вхідного каналу розміщена абсолютна адреса, яка починається з символу «%». Далі в адресі вказується область пам'яті ПЛК (**I** – область пам'яті входів, **Q** – області пам'яті виходів), розмір пам'яті (**X** – біт, **B** – байт, **W** – два байти або слово, **D** – чотири байти або подвійне слово) та зсув адреси відносно початкового значення адреси, тобто від нульового байта. Далі після символу «:» вказується тип даних, який відповідає розміру пам'яті для каналу. Далі йде коментар, який обмежений символами (*.....*).

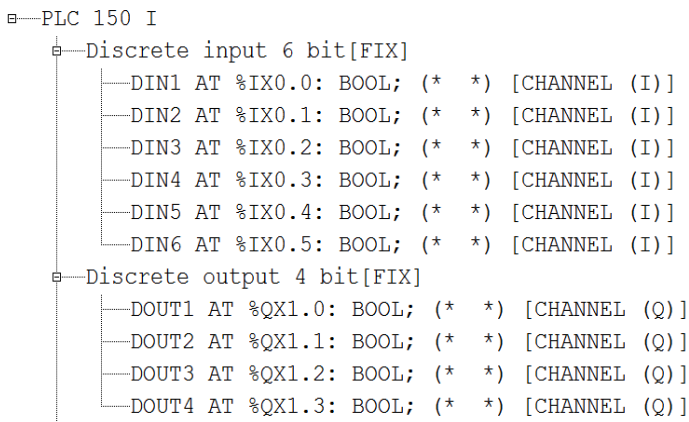


Рис. 1.8 – Вікно з відкритими слотами вхідних та вихідних каналів ПЛК

В ПЛК є можливість встановлення режиму роботи дискретних входів. Вони можуть бути налаштовані на визначення стану або на визначення фронтів. Це означає, що в першому випадку процесор в ПЛК визначає стан входу типу «сухий контакт» (замкнено або не замкнено) один раз на кожному циклі сканування, а в другому випадку – вхід налаштований на імпульсний режим роботи, тобто процесор оцінює зміну стану входу з незамкненого на замкнений. Чутливість та швидкість реакції ПЛК на зміну стану входу визначається встановленням часу фільтрування (Time of filtration) окремо для кожного дискретного входу. Для імпульсного режиму цей параметр дорівнює значенню -1, для звичайного режиму – будь-яке значення з можливого діапазону 0...10000. Вибір цього параметра залежить від умов роботи установки. Але для швидкоплинних процесів він не повинен бути занадто великим, що призведе до втрати частки поточних даних від датчика. Напроти, для повільних процесів – занадто малим, що призведе до перевантаження процесора ПЛК. Дискретні виходи ПЛК мають можливість налаштування безпечного стану (параметр Safe Value) вихідних пристроїв у випадку аварії.

Взагалі, в імпульсному режимі до дискретних входів можуть бути підключені інші датчики, крім датчика типу «сухий контакт». Це, наприклад, безконтактний датчик наближення, фотодатчик або енкодер. За допомогою цих датчиків можливо вирішення більш складних завдань, наприклад, лічення продукції на конвеєрі за допомогою фотодатчика, визначення швидкості (частоти обертів) конвеєра за допомогою безконтактного датчика наближення або визначення кутового положення вала привода конвеєра за допомогою енкодера. Для таких режимів роботи дискретних входів за допомогою контекстного меню в конфігурацію ресурсів ПЛК можна додати змінні програмні модулі Trigger, Counter, Encoder, тощо. Ці модулі розширюють функціональні можливості апаратних ресурсів ПЛК. В цьому випадку звичайний дискретний вхід (Channel) деактивується. Тобто, цей вхід переходить в спеціальний режим роботи. Тому, в програмному модулі потрібно вказати номер фізичного каналу, з яким з'єднаний датчик.

Додатково в ПЛК є можливість формування ШІМ-сигналів на дискретних виходах ПЛК. Докладно це буде розглянуто далі в третьому завданні при вивченні законів регулювання. Ознайомитися з порядком налаштування дискретного виходу на режим роботи ШІМ-регулювання можна в джерелах [5, 6].

Для конфігурування аналогових входів ПЛК в його ресурсах є чотири однакових слоти **Unifed signal sensor [SLOT]**. Кожний слот відповідає одному фізичному аналоговому входу ПЛК та вміщує два канали: власне вимірювальний (типу **REAL** з коментарем **Value**) та канал визначення часу вимірювання (типу **WORD** з коментарем **Circular time**). Перший канал потрібен для зберігання поточного значення параметра, а другий – для обчислення сигналу регулювання в процесі реалізації ПІД-регулювання.

Слоти аналогових входів можуть бути налаштованими на оброблення сигналів від джерел сигналу різного типу: уніфікованого датчика сигналу (встановлений за умовчанням), термометра опору, термоелектричного перетворювача (термопара) та контактного датчика («сухий контакт»). Змінити тип джерела можна за допомогою контекстного меню. Для цього ЛКМ слід помітити потрібний слот конфігурації ресурсів ПЛК. В результаті він буде позначений рамкою з точок. Далі для виклику контекстного меню натисніть ПКМ та у переліку, що випадає, виберіть потрібний тип джерела сигналу, як це показано на рис. 1.9. На цьому рисунку показано порядок заміни уніфікованого датчика сигналу (**Unifed signal sensor**) на термопару (**Termocouple sensor**). Подібно до написаного раніше можливо змінити уніфікований датчик сигналу (**Unifed signal sensor**) на термоперетворювач опору (**RTD sensor**). Зауважимо, що розташування слотів аналогових входів в дереві конфігурації повністю відповідає номеру фізичного входу ПЛК.

Для кожного типу джерела в першому рядку параметрів – *Параметри модуля* – можна вказати конкретний тип первинного перетворювача сигналу. Для уніфікованого датчика сигналу це може бути уніфікований струмовий сигнал (за умовчанням це $0 \dots 20 \text{ mA}$), сигнал напруги або резистивний сигнал. Якщо в якості джерела сигналу був вибраний термометр опору, то в переліку датчиків доступні платинові (за умовчанням встановлений тип ТСП-50, з НСХ, у якої коефіцієнт дорівнює 1.385), мідні та інші перетворювачі з різними НСХ. Для налаштування на оброблення сигналу від термопар також можливий вибір їхнього типу (за умовчанням встановлений тип термопари – ТХК (L)). В [5] наведено всі типи джерел сигналів з їхніми позначеннями в конфігурації па-

раметрів ресурсів ПЛК. Якщо потрібно збільшити кількість дискретних входів в ПЛК, то це можна зробити за рахунок аналогових входів. Для цього у відповідному слоті замініть аналоговий перетворювач на контактний датчик типу «сухий контакт» – **Contact sensor** (див. рис. 1.9). Отже, розглянуто основні правила роботи з дискретними входами/виходами та аналоговими входами контролера.

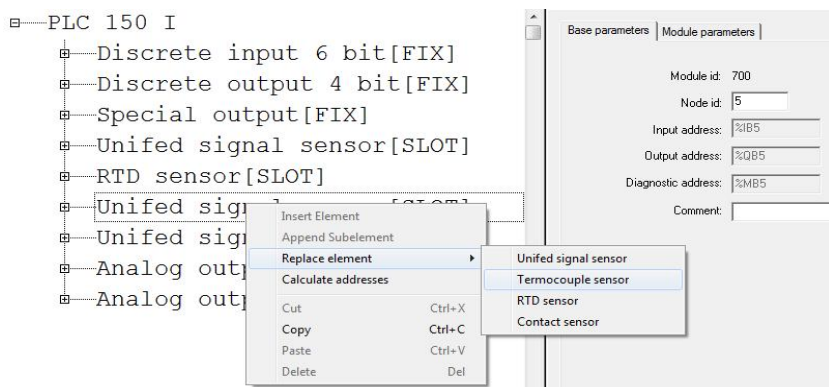


Рис. 1.9 – Порядок зміни типу джерела сигналу в слоті ПЛК

Розглянемо інші слоти фіксованого набору ресурсів ПЛК (див. рис.1.7). Це спеціальний слот з ім'ям **Special output (FIX)** та слоти аналогових виходів з ім'ям **Analog output (FIX)**. Модуль спеціального дискретного виходу містить бітовий канал для керування вбудованим звуковим випромінювачем або світлодіодом і не має фізичних клем на зовнішніх колодках ПЛК. Якщо значення каналу **TRUE**, то вмикається випромінювач або світлодіод. Аналогові виходи відносяться до фіксованих модулів ПЛК. Залежно від типу встановлених в ПЛК ЦАП існують три варіанти налаштування модуля аналогових виходів:

- універсальний, який програмно налаштується або на ЦАП «*параметр-струм*», або на ЦАП «*параметр-напруга*»;
- фіксований, призначений для ЦАП «*параметр-струм*»;
- фіксований, призначений для ЦАП «*параметр-напруга*».

Про тип ЦАП в ПЛК свідчить літера в його моделі. Для ЦАП типу «*параметр-струм*» це літера «**I**», для ЦАП типу «*параметр-напруга*» – «**U**» та для універсального ЦАП – літера «**A**».

На стендах встановлено ПЛК150.И-L з виходами типу «струм 4...20 мА», які неможливо замінити на інші для даної моделі ПЛК. Тобто в розпорядженні розробника СПЗ є лише струмовий вихід.

Таким чином, виходячи з принципової електричної схеми стенда (див. рис. 1.2) налаштуйте аналогові входи для отримання сигналів від первинних перетворювачів. Так, слот другого аналогового входу налаштуйте на отримання сигналу від термоперетворювача опору з НСХ Pt100. Третій слот налаштуйте на отримання сигналу від термопари типу К. Четвертий слот налаштуйте на вимірювання пасивного сигналу опору від змінного резистора номіналом 0...1 кОм. Зробіть налаштування вхідних каналів відповідно до екранних форм, які зображені на рис. 1.10, рис. 1.11 та рис.1.12.

Зверніть увагу на особливості налаштування аналогового входу для оброблення сигналу від резистивного датчика. В рядку налаштування значення верхньої межі *Ain high* вкажіть число 5000. Це потрібно для погодження діапазону зміни реального резистора (0...1 кОм) з типом програмного АЦП – *RO_5000* в першому рядку параметрів модуля. В такому випадку значення змінної *AIN4* буде збігатися з реальним діапазоном значень резистора, тобто 0...1 кОм.

Якщо вказати інше значення верхньої межі, наприклад, на рівні 1000 од., то значення змінної буде знаходитись в діапазоні 0...200 од. (коефіцієнт перетворення дорівнює п'яти). Це налаштування має назву «масштабування» параметра і може бути використане для перетворення параметра в відсоткове значення безпосередньо в АЦП.

Насамкінець, налаштуйте аналоговий вихід ПЛК АО1, надавши йому символічне ім'я *AOUT1*. Цей вихід потрібен для формування управляючого сигналу для виконавчого механізму, наприклад, для мембранного виконавчого механізму, який через електричний пневматичний перетворювач з'єднаний з ПЛК. В якості програмного задатчика сигналу керування будемо використовувати змінний резистор, який підключений до входу *A14* ПЛК. Але для цього потрібно розробити відповідну програму перетворення поточного значення резистора в уніфікований струмовий сигнал 4...20 мА. Додатково розробіть логічну програму управління дискретними виходами ПЛК залежно від стану дискретних датчиків на його входах. Це буде зроблено на наступному етапі виконання завдання.

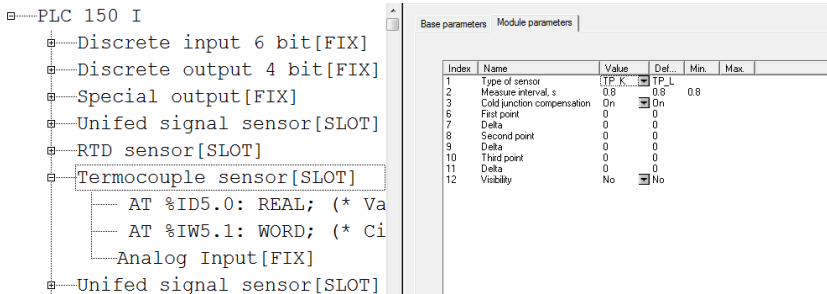


Рис. 1.10 – Вікно налаштування аналогового входу для оброблення сигналу від термоперетворювача опору

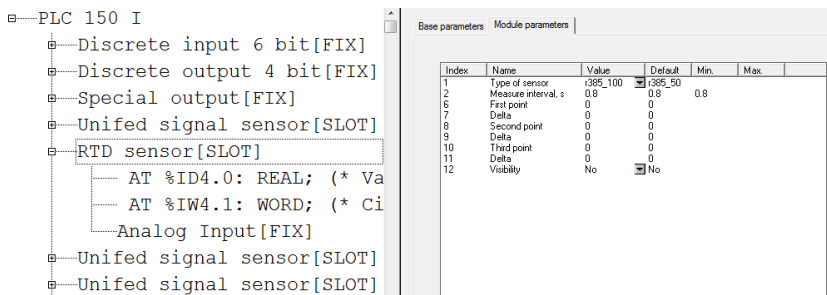


Рис. 1.11 – Вікно налаштування аналогового входу для оброблення сигналу від термопари

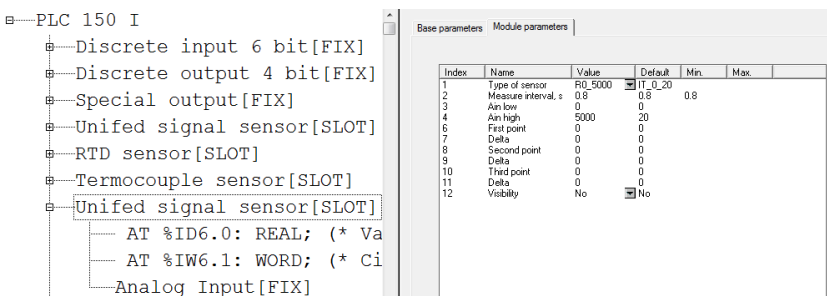


Рис. 1.12 – Вікно налаштування аналогового входу для оброблення сигналу від змінного резистора

1.4.3. Розроблення програми користувача

Поверніться з вікна налаштувань конфігурації ресурсів ПЛК до вікна редактора *POU*. Для цього в менеджері проекту, що знаходиться ліворуч (див. рис.1.6), натисніть ЛКМ по вкладці *POU*, а далі – двічі по головному *POU* з ім'ям *PLC_PRG* – для його відкриття для редагування.

Логічна програма для управління дискретними входами повинна працювати таким чином: виходячи з комбінацій логічних умов, потрібно вмикати або вимикати вихідні прилади, які підключені до ПЛК. Також потрібно вимірювати температуру за допомогою термоперетворювача опору та термопари, а також вимірювати опір змінного резистора. Причому в програмі користувача необхідно перетворювати поточне значення резистора в уніфікований струмовий сигнал 4...20 мА. Також поточне значення опору змінного резистора потрібно перетворити у значення положення клапану в одиницях відсотків від 0 до 100 %.

Отже, програма користувача буде виглядати так, як це показано на екранній формі (див. рис. 1.13). Процес розроблення програми полягає у виборі ЛКМ потрібних інструментів (входів, виходів, блоків, тощо) та перенесенні їх до робочого простору редактору з подальшим з'єднанням вибраних елементів. Редактор мови *CFC* дозволяє вільне розміщення елементів, але на етапі ознайомлення з ним вважається доцільним розміщати елементи рядками. Перед компілюванням проекту потрібно виконувати команду **Order everything according to data flow**, яка викликається ПКМ і потрібна для визначення порядку виконання елементів програми згідно з потоком даних. Порядок виконання програми визначається числом у верхньому правому куті елемента.

Після створення програми користувача знову збережіть проект. Далі проведіть його компілювання, вибравши ЛКМ команду **Build** або **Rebuild all** в меню **Project** або натиснувши на функціональну клавішу **F11**. Але ця функціональна клавіша відповідає лише першій команді. Різниця між командами полягає у тому, що перша команда виконується лише для активного вікна з відкритим *POU*. Виконання другої команди призводить до компілювання всього проекту. Якщо в програмі немає помилок, то компілювання пройде успішно. У цьому випадку проект буде готовий до завантаження до ПЛК. Про це буде свідчити стрілка синього кольору напроти відповідних *POU* в менеджері проекту.

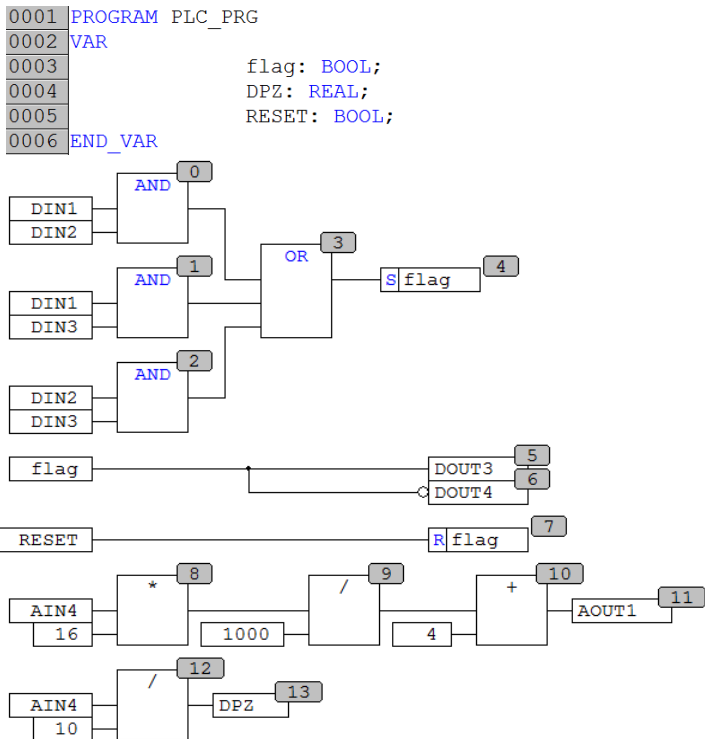


Рис. 1.13 – Фрагмент програми користувача

Далі можна діяти двома шляхами: або налагодити проект в режимі емуляції ПЛК, або завантажити проект до реального ПЛК. У першому випадку достатньо в меню **Online** ЛКМ поставити покажчик поряд з рядком **Simulation Mode**, а потім вибрати команду **Login**. Після завантаження проекту до емулятора потрібно запустити виконання програми користувача командою **Run**, або натиснувши на функціональну клавішу **F5**. Одразу в меню **Online** стануть доступними команди для налагодження проекту, котрі зображені на рис. 1.14. Зауважимо, що в рядку статусу, який знаходиться внизу вікна (див. рис. 1.6), буде відображатися стан зв'язку середовища *CDS2* та ПЛК – **ONLINE**, режим виконання програми користувача – **SIM** та статус емулятора – **RUNNING**.

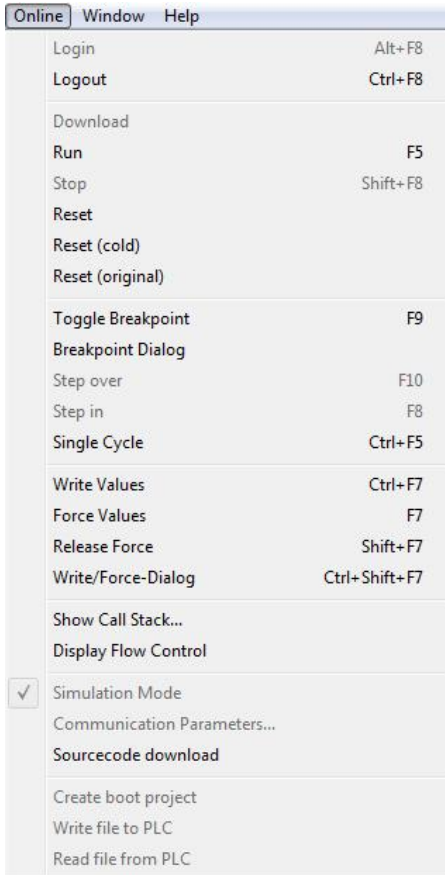


Рис. 1.14 – Вікно з командами меню Online

Виконання програми користувача на емуляторі має деякі незручності. В першу чергу це стосується присвоєння значень змінним в програмі користувача. Ця дія виконується за два кроки. Спочатку потрібно ЛКМ в програмі визначити потрібну змінну та ввести потрібне значення. Потім за допомогою команди **Write Values** (або гарячих клавіш **Ctrl+F7**) передати це значення до ПЛК (див. рис. 1.14).

Для того, щоб перевірити правильність роботи програми користувача в середовищі *CDS2* є так званий відладчик. Це утиліта, яка дозволяє налаштувати точки зупинення програми в потрібних місцях програми та в покроковому режимі перевірити правильність виконання програми та реакцію об'єкта керування на введення значень до змінних.

Перед запуском режиму перевірки програми користувача потрібно встановити точки зупинення програми за допомогою команди **Breakpoint Dialog** в меню **Online** (див. рис. 1.14). Після активування цієї утиліти з'явиться вікно **Breakpoints**, яке зображене на рис. 1.15. Далі необхідно визначити потрібні точки зупинення програми користувача та підтвердити свій вибір натисненням ЛКМ на кнопку **ОК**. Важливо пам'ятати, що ця утиліта запускається лише тоді коли встановлений зв'язок між середовищем *CDS2* та ПЛК,

тобто режим зв'язку – **ONLINE**. Але після переходу в режим **OFFLINE** (команда **Logout**) точки зупинення втрачаються. Якщо потрібно налагодити проект знову, необхідно відкрити діалогове вікно для створення точок зупинення.

На рис. 1.14 зображені усі команди та гарячі клавіші для їхнього виклику, які використовують в процесі налагодження проекту. Також в меню **Online** присутні команда для створення коду для завантаження в енергонезалежну пам'ять ПЛК (**Create boot project**) та команда для завантаження цього коду (**Sourcecode Download**).

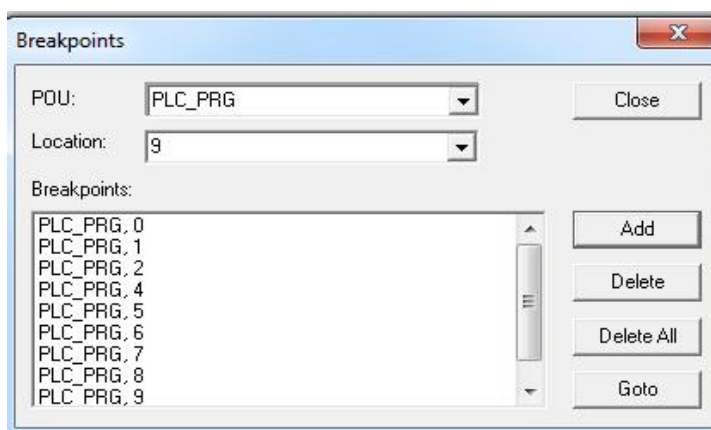


Рис. 1.15 – Вікно Breakpoints

Перехід в режим **OFFLINE** здійснюється за допомогою команди **Logout** (див. рис. 1.14) в меню **Online**, або за допомогою відповідних гарячих клавіш (**Ctrl+F8**). Для подальших дій необхідно зняти покажчик напроти рядка **Simulation Mode**.

1.4.4. Підключення CDS2 до ПЛК та завантаження до нього коду проекту

Завантаження готових проектів до ПЛК можливе за допомогою двох інтерфейсів: послідовного – **RS-232** та локальної мережі **Ethernet** з протоколом **TCP/IP**. Після компілювання програми за допомогою відповідного програмного комунікаційного модуля з'єднайтесь з ПЛК та за-

вантажте код проекту до ПЛК. Для цього необхідно відкрити вікно **Communication Parameters** та налаштувати параметри програмного комунікаційного модуля. Натисніть ЛКМ на команду **Communication Parameters...** (див. рис. 1.14) в меню **Online**. Відкриється вікно, яке зображене на рис. 1.16. Натисніть ЛКМ на кнопку **New...** для відкриття вікна налаштування параметрів нового каналу зв'язку **Communication Parameters: New Channel** з переліком можливих драйверів каналів зв'язку з ПЛК. Розглянемо обидва варіанти підключення: через послідовний інтерфейс **RS-232** та через **Ethernet**.

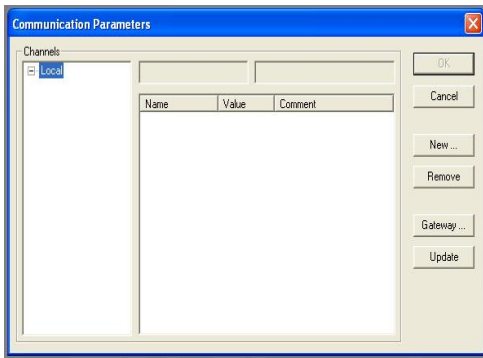


Рис. 1.16 – Вікно налаштування модуля *Communication Parameters*

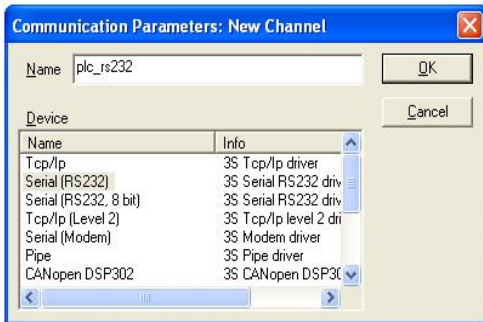


Рис. 1.17 – Вікно вибору драйвера для каналу зв'язку з ПЛК

Для з'єднання за послідовним інтерфейсом через порт ПЛК *Debug RS-232* в переліку **Device** серед доступних драйверів виберіть **Serial (RS232)**. До поля **Name** введіть ім'я каналу, наприклад, `plc_rs232`. В результаті буде отримане вікно, яке зображене на рис. 1.17. Натисніть ЛКМ на кнопку **OK**. Далі відкриється наступне вікно для налаштування параметрів **COM**-порту ПК. Введіть потрібні дані в поля рядків з назвами параметрів стовпчика **Value**. Для зміни параметра слід двічі клікнути ЛКМ по наявному значенню параметра та, перегортаючи список доступних значень стрілками на клавіатурі ПК, вибрати потрібне значення. Для

вводу нового значення натисніть клавішу **Enter** на клавіатурі. Отже, вкажіть номер *COM*-порту, до якого фізично підключений ПЛК та швидкість передавання даних. За умовчанням після створення нового каналу встановлена швидкість буде дорівнювати 38400 біт/с. Але, для завантаження коду проекту передбачена лише одна швидкість – 115200 біт/с. Змініть встановлену швидкість на потрібну, тобто, на 115200 відповідно екранній формі, яка зображена на рис. 1.18. Інші параметри залишити без змін. Завершіть налаштування каналу зв'язку натисненням ЛКМ на кнопку **OK**.

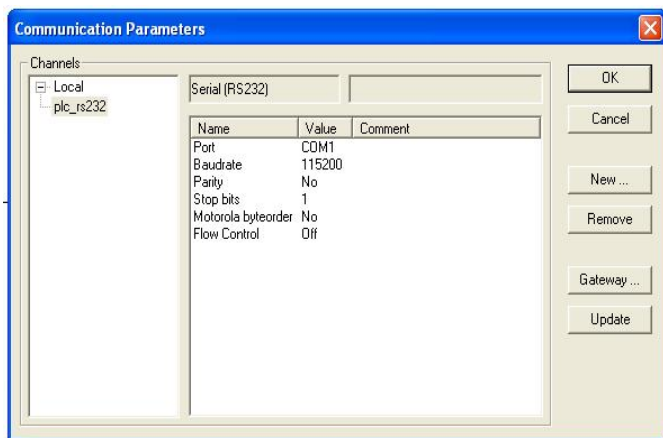


Рис. 1.18 – Вікно налаштувань параметрів *COM*-порту

Для налаштування зв'язку з ПЛК по *Ethernet*-каналу потрібно у вікні **Communication Parameters: New Channel** (див. рис. 1.17) вибрати рядок з драйвером *Tcp/Ip (Level 2)*. Далі натисніть ЛКМ на кнопку **OK**. Відкриється вікно **Communication Parameters** для налаштування параметрів *Ethernet* з'єднання. Його вигляд буде подібним до екранної форми, яка зображена на рис. 1.19.

Зауважимо, що налаштування зв'язку з використанням *Ethernet* з'єднання за протоколом *TCP/IP* має деякі особливості. Це стосується заповнення першого рядка з ім'ям **Address**, де вказується *IP*-адреса ПЛК замість значення *localhost* в стовпчику **Value**. Інші параметри залишити без змін.

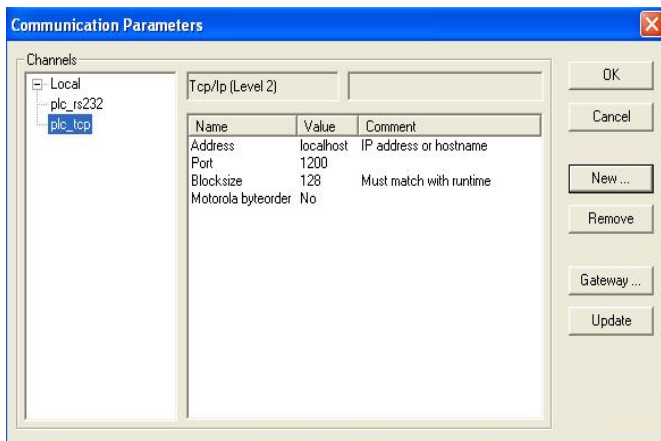


Рис. 1.19 – Вікно налаштувань параметрів *Ethernet* з'єднання

Важливо знати, що усі нові ПЛК мають одну адресу – 10.0.6.10, яка встановлюється під час їхнього виробництва на заводі. Для подальшого використання ПЛК можна змінити його адресу на іншу, яка буде відповідати сегменту локальної мережі. Зміна *IP*-адреси контролера можлива за допомогою команди **SetIP ***.***.***.*****, яка встановлюється через браузер ПЛК в ресурсах контролера в середовищі *CDS2*. При цьому попередньо має бути встановлений зв'язок з контролером через інтерфейс *Debug RS-232*. Крім того, можлива зміна *IP*-адреси за допомогою програми *EasyWorkPLC.exe*, яка знаходиться в директорії зі шляхом доступу *C:\Program Files\Owen\EasyWorkPLC*.**. Проте *MAC*-адреса ПЛК є оригінальною і не може бути зміненою. Її значення можна визначити на позначці, що наліплена збоку ПЛК праворуч або програмно за допомогою браузера (*GetIP*) в середовищі *CDS2*.

Для налаштування каналу зв'язку з ПЛК за адресою 10.0.6.10 потрібно, щоб контролер та комп'ютер знаходились в одній локальній підмережі та фізично з'єднані *cross-over* кабелем. При цьому мережний адаптер ПК може мати будь-яку адресу, крім адреси 10.0.6.0, 10.0.6.10 та 10.0.6.255. Для налаштування параметрів мережного адаптера ПК натисніть ЛКМ послідовно на програмні кнопки *ПУСК*, виберіть *Сетевое окружение* і виберіть команду *Отобразить сетевое окружение* в середовищі *Windows*. Далі активуйте вікно налаштувань властивостей

локального з'єднання *Протокол Інтернет (TCP/IP)* та введіть адресу ПК (яка відрізняється від адреси ПЛК, але входить до його сегмента, наприклад 10.0.6.11) та маску (255.255.255.0) у вікні *Додатково*.

Якщо ПЛК та ПК знаходяться в одній мережі і не мають безпосереднього з'єднання, тобто для зв'язку використовується мережне обладнання, в рядок **Address** замість **localhost** ведіть адресу ПЛК. Для збереження нового значення *IP*-адреси натисніть на клавішу **Enter** на клавіатурі ПК та завершіть налаштування каналу зв'язку натисненням ЛКМ на кнопку **OK** (див. рис. 1.19).

Після налаштування з'єднання виберіть команду **Login** в меню **Online**, яка встановлює зв'язок з контролером. При цьому покажчик напроти рядка меню **Simulation Mode** має бути знятим. На рис. 1.20 зображено фрагмент програми користувача в режимі **ONLINE** та **RUNNING**.

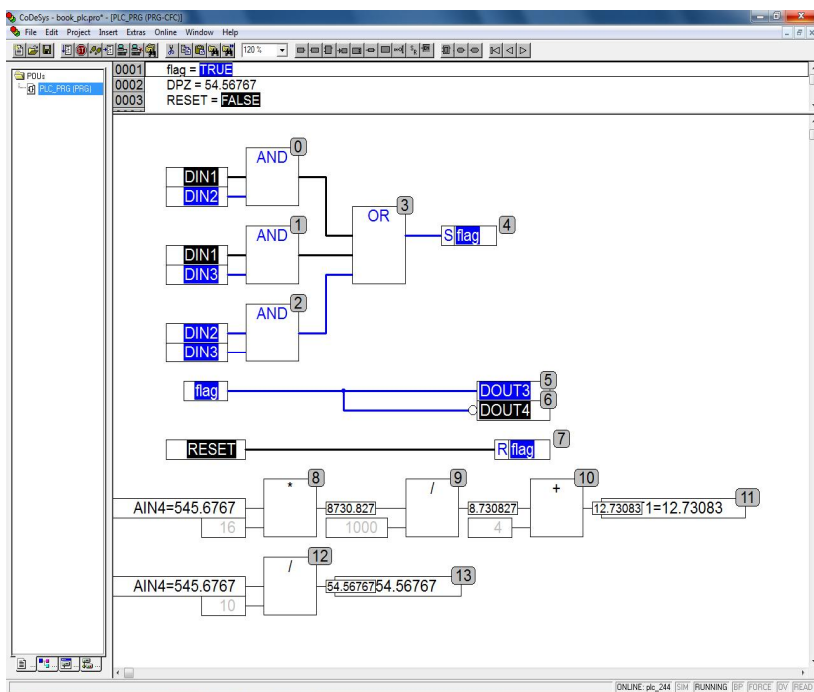


Рис. 1.20 – Програма користувача в режимі **ONLINE** та **RUNNING**

УВАГА! Для зміни інтерфейсу з'єднання необхідно провести перезавантаження ПЛК, натиснувши на «втоплену» кнопку *Сброс* на його передній панелі. Для цього використовуйте, наприклад, скрепку або викрутку.

1.4.5. Створення візуалізації проекту

В середовищі *CDS2* для налагодження проекту та зручного спостереження за змінними потрібно створити його візуалізацію. Натисніть ЛКМ на вкладення *Visualizations* у менеджері об'єктів для переходу та відкриття редактора візуалізацій (див. рис. 1.6). Далі, ПКМ викличе контекстне меню та виберіть команду **Add Object ...** для створення нового об'єкта **New Visualization**. Надайте ім'я цій візуалізації, наприклад, *PLC_VISU* (в *CDS2* дозволено використовувати лише латинські літери та цифри), та введіть цю назву до рядка **Name of the new Visualization**. Підтвердить створення візуалізації натисненням ЛКМ на кнопку **OK**. В результаті з'явиться вікно з робочим простором для розроблення візуалізації та панеллю інструментів для створення графічних елементів та об'єктів. За допомогою різних інструментів створіть графічні елементи з відповідними налаштуваннями їхніх властивостей.

Основна концепція налаштування графічних елементів та об'єктів полягає в наступному. Спочатку вибирається потрібний графічний елемент. Далі визначається поведінка цього елемента в залежності від значення змінної. Насамкінець, до графічного елемента прив'язується потрібна змінна з програми користувача. Інші налаштування не мають принципового значення та визначають лише зовнішній вигляд елемента.

Програма користувача, яка зображена на рис. 1.13, складається з декількох ланцюгів. Ці ланцюги є умовами та діями над змінними, які мають локальний або глобальний статус. Так, оператори з номерами від 0-го до 4-го формують статус змінної *flag* типу **BOOL**. Змінна *flag* отримує статус **TRUE**, якщо одночасно в стані **TRUE** будуть будь-які два перемикачі з трьох: *DIN1*, *DIN2* та *DIN3*. В залежності від статусу змінної *flag* визначається статус фізичних виходів *DOUT3* та *DOUT4*. Причому вихід *DOUT3* буде в стані **TRUE**, якщо змінна *flag* отримує

статус **TRUE**. Відповідно вихід **DOUT4** буде в протилежному стані, тобто в стані **FALSE** (використаний символ інверсії на вході). Змінна **RESET** потрібна для скидання статусу змінної **flag** після вимикання дискретних входів. Групи операторів з 8-го до 11-го та 12 й 13 реалізують перетворення значення змінного резистора **AIN4** у вихідний сигнал постійного струму **AOUT1** (4...20 **mA**) та в змінну **DPZ**, яка відображає положення клапана у відсотках. В програмі користувача використані одночасно глобальні та локальні змінні. Символьні імена фізичних каналів ПЛК автоматично є глобальними, а локальні знаходяться в переліку об'явлення змінних відповідного **POU**. Різниця між глобальними та локальними змінними полягає в тому, що локальні змінні доступні в межах лише одного **POU**, а глобальні – в межах усього проекту.

Зауважимо, що в цілому приклад має навчальну мету та відображає принципи розроблення програми користувача та її візуалізації.

Отже, у вікні візуалізації створіть графічні елементи для відображення значень та стану змінних. В редакторі візуалізації є декілька груп інструментів для створення візуалізацій. По-перше, це так звані примітиви – **Rectangle** (Прямокутник), **Rounded Rectangle** (Прямокутник з круглими кутами), **Ellipse** (Еліпс), **Polygon** (Багатокутник), **Polyline** (Ломана лінія), **Curve** (Крива), **Pie** (Сектор). По-друге, це графічні об'єкти – **Bitmap** (Растровий рисунок), **Visualization** (Візуалізація), **Button** (Кнопка), **Table** (Таблиця), елемент **ActiveX**, **Meter** (Стрілочний індикатор), **Bar Display** (Показуючий стовпчик), **Scrollbar** (Слайдер), **Histogram** (Гістограма), **Alarm table** (Таблиця тривоги), **Trend** (Тренд), **WMF file** (файл у форматі **WMF**). Більш докладно з порядком налаштування цих графічних елементів та об'єктів можна ознайомитись за допомогою довідкової системи у відповідному розділі. Зараз покажемо налаштування деяких графічних елементів відповідно до змінних в програмі користувача.

Отже, створіть елемент **Ellipse** для відображення стану дискретного входу або виходу ПЛК, а також статусу локальної змінної **flag**. Проведіть налаштування властивостей створеного елемента. Для цього підведіть покажчик миші до елемента (допоки стрілка не перетвориться в долонь) та натисніть ЛКМ по ньому. В результаті він буде позначений контуром зі знаком «+» в центрі. За допомогою контекстного меню (ПКМ), виберіть

команду **Configure**. Відкриється вікно **Regular Element Configuration**, екранна форма якого зображена на рис. 1.21. Як зображено на рис. 1.21 в категорії **Shape** можлива швидка зміна типу примітиву на інший.

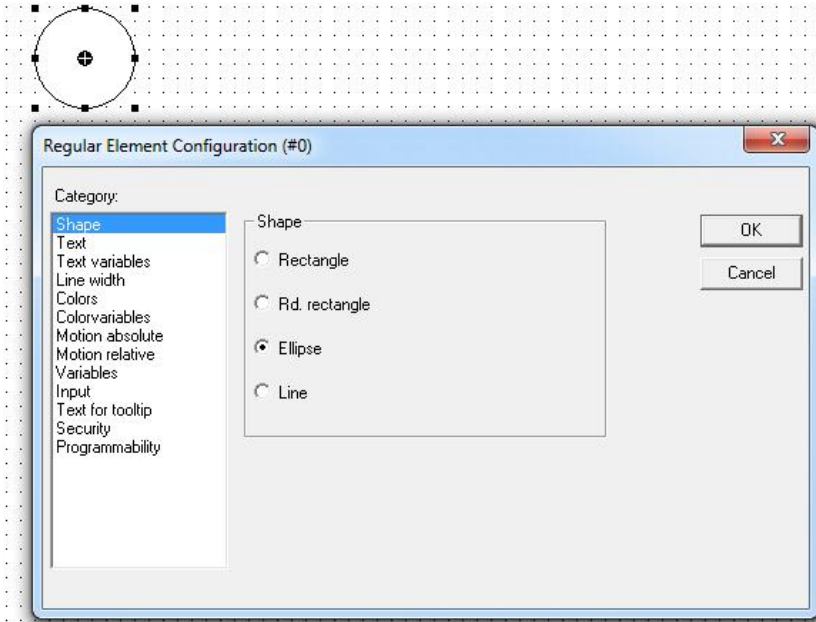


Рис. 1.21 – Вікно налаштувань параметрів графічного елемента **Rectangle**

Відкрийте категорію **Text** та введіть в поле **Content** наступний текст: **SENSOR 1**. В цій категорії можливе налаштування властивостей введеного тексту, які є стандартними для будь-яких текстових редакторів, наприклад *Microsoft Word*. Далі відкрийте категорію **Colors**, в якій налаштуйте колір фону елемента залежно від значення змінної, яка буде зв'язана з ним. Наприклад, для нормального стану змінної **DIN1**, тобто **FALSE**, натисніть на кнопку **Inside** в полі **Color**, а у відкритому вікні палітри кольорів виберіть сірий колір та підтвердіть свій вибір натисненням на кнопку **OK**. Для аварійного стану змінної **DIN1**, тобто **TRUE** в полі **Alarm color** також натисніть на кнопку **Inside** та виберіть червоний колір. Далі, в категорії **Variables** необхідно вказати ім'я змінної, від

значення якої буде залежати колір графічного елемента **Ellipse**. Для цього ЛКМ поставте покажчик миші в рядок з назвою **Change color:** . Далі введіть ім'я змінної в такому форматі «.din1» або використайте **Input Assistant**, який викликається функціональною клавішею **F2**. У відкритому вікні асистента клікніть по символу «+» напроти списку **Globale_Variables** і далі виберіть потрібну змінну (**DIN1**). Подібним чином створіть та налаштуйте інші графічні елементи для відображення стану дискретних входів, виходів та локальної змінної **flag**. Остання змінна знаходиться в переліку **PLC_PRG (PRG)**, тобто належить головному **POU**.

Далі створіть графічний елемент **Rectangle** для відображення значення температури всередині теплового об'єкту, тобто змінної **AIN2**. Налаштування властивостей графічного елемента **Rectangle** подібне до налаштувань графічного елемента **Ellipse**. Але є деякі відмінності, які додають функцію відображення числового значення.

По-перше це відноситься до налаштувань категорії **Text**. Для відображення статичного тексту в поле **Content** введіть назву параметра – **Temperatura=**. Далі за допомогою одночасного натиснення на гарячі клавіші **Ctrl+Enter** зробіть перехід до другого рядка у полі **Content**. В другому рядку введіть динамічну складову тексту – **%3.1f °C**. Цей запис має назву **placeholder** (заповнювачем), який визначає формат відображення змінної. Символ «%» в цьому запису вказує на початок формату відображення заповнювача, символи «3.1» – вказують на кількість розрядів у відображенні цілої та дробової частини числа та символ «f» – вказує на тип даних змінної, значення якої відображається (в даному випадку це число з плаваючою комою, тобто тип даних **REAL**). Інші позначення визначають одиницю виміру параметра, який відображається. Наприклад, якщо потрібно вказати одиницю виміру температури у градусах Цельсія, то це будуть символи «° C». Знак градуса «°» при цьому вводиться за допомогою гарячих клавіш з таблиці символів операційної системи **Windows** (версії від **XP** до **7**). Це одночасне натиснення на клавішу **Alt** та цифри **0, 1, 8 та 6** на клавіатурі ПК (тобто, **Alt+0186**).

Для відображення змінних інших типів даних використовують такі символи: для десяткових чисел – символи «d» або «i», для десяткових

чисел без знака – «U». Взагалі, будь-який тип даних можна відобразити, якщо використати символ «S», який вказує на неявне перетворення будь-якого типу в тип даних **STRING**. В такому випадку значення змінної відображається як послідовність символів значення змінної. Зауважимо, що символ типу даних в заповнювачі повинен бути обов'язково строкового регістру. В *CDS2* є можливість відображення часу та дати. Для цього використовують спеціальний заповнювач %t і далі набір заповнювачів, які визначають формат виведення часу та дати (див. довідку до *CDS2*).

Завершує налаштування графічного елемента **Rectangle** його зв'язування зі змінною, яка буде відображатися. Для цього в категорії **Variables** за допомогою асистента введення у поле **Textdisplay**: потрібно вставити символічне ім'я другого фізичного каналу аналогового входу – **AIN2**. Графічний елемент **Rectangle** можна зробити активними, тобто додати можливість введення значення в змінну. Для цього в категорії **Input** необхідно вибрати потрібну дію. Це можуть бути такі дії: присвоєння змінній типу **BOOL** стану **TRUE** або **FALSE** в режимі **Toggle variable** («перемикач») або **Tap variable** («кнопка»), присвоєння змінній чисельного або строкового типу потрібного значення за допомогою екранних форм **Text input of variable 'Textdisplay'** для введення літер або цифр, а також для виконання переривань **Execute program**: або переходу до інших візуалізацій **Zoom to vis.:** . Вибір потрібної дії визначається встановленням позначки напроти її в рядку. На рис. 1.22 зображено вікно налаштування параметрів графічного елемента **Rectangle** з категорією **Input**. Таким чином, примітивні графічні елементи є найбільш зручними для створення візуалізацій з точки зору кількості можливих функцій. Інші категорії налаштування параметрів графічного елемента **Rectangle** лише розширюють їхню функціональність.

Створить графічний елемент **Button** для скидання стану змінної **flag**, яка є результатом аналізу стану датчиків **DIN1**, **DIN2** та **DIN3**. На рис. 1.23 зображено вікно налаштувань категорії **Bitmap** графічного елемента **Button**. Цей графічний елемент **Button** має менші можливості щодо налаштувань відносно графічного елемента **Rectangle**. Тобто, функція введення стану до змінної вже є в цьому елементі. Достатньо зв'язати цій елемент з потрібною змінною. Тому, в категорії **Input** введіть до поля **Tap variable** ім'я потрібної змінної. Це буде локальна змінна **PLC_PRG.RESET**.

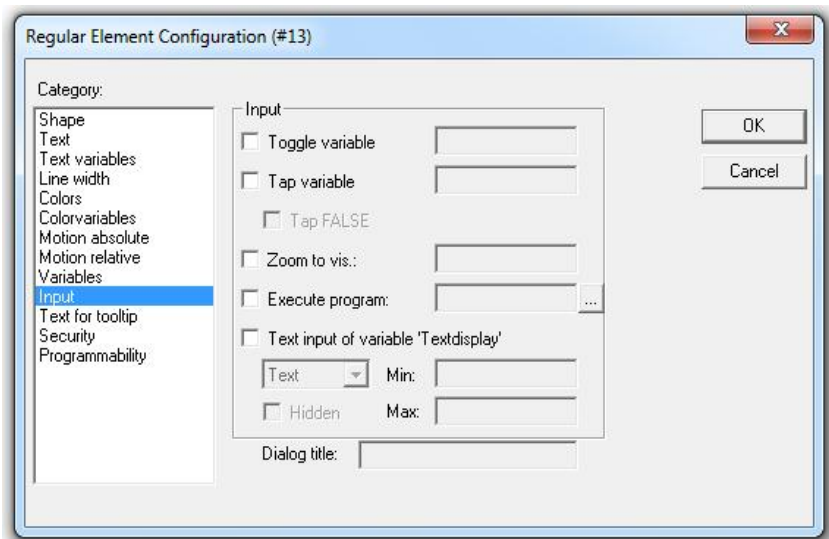


Рис. 1.22 – Вікно налаштувань категорії Input графічного елемента Rectangle

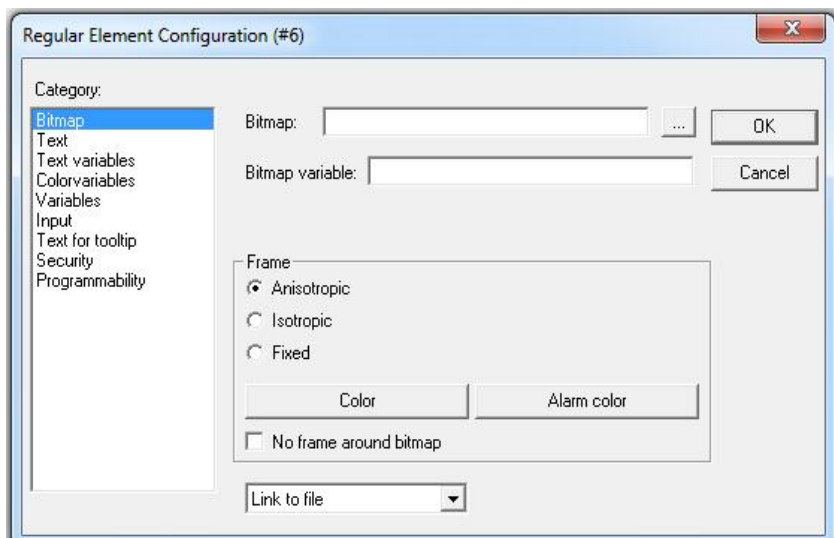


Рис. 1.23 – Вікно налаштувань категорії Bitmap графічного елемента Button

ПРИМІТКА. Якщо з графічним елементом зв'язується локальна змінна, то перед нею обов'язково вказується *POU*, в якому вона створена.

Інші графічні елементи виконують функцію припинення роботи програми користувача та функцію переходу до інших візуалізацій. Також візуалізація має статичні графічні елементи, тобто текстові рядки. Всі вони використовують властивості графічного елементу **Rectangle**.

Отже, створіть візуалізацію з ім'ям **PLC_VISU** подібно до зображеної на рис. 1.24.

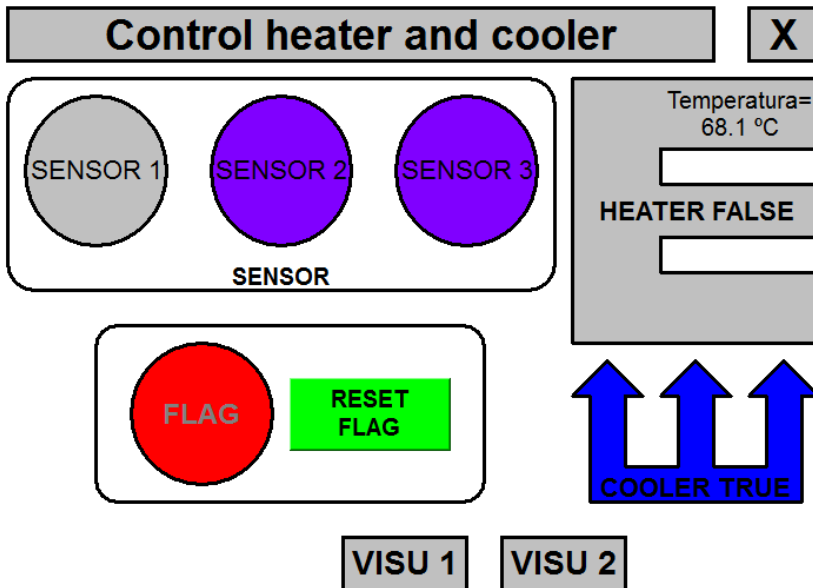


Рис. 1.24 – Приклад візуалізації з ім'ям **PLC_VISU**

Додайте до проекту ще дві візуалізації з ім'ям **VISU_1** та **VISU_2**, які будуть подібними до зображених на рис. 1.25 та 1.26. В даних візуалізаціях передбачте кнопки для повернення до візуалізації з ім'ям **PLC_VISU**. Також в цих візуалізаціях створить інші графічні об'єкти, наприклад, **Trend**, **Meter** та **Bar Display**. Ці об'єкти є більш складними ніж графічні елементи, але їхнє налаштування буде подібним до розглянутого раніше. Тобто, потрібно налаштувати їхні властивості та вказати змінні, які пов'язані з цими об'єктами.

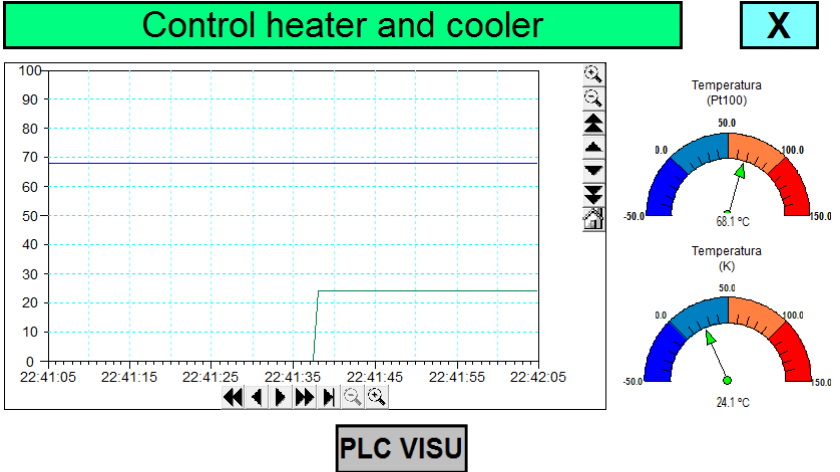


Рис. 1.25 – Приклад візуалізації з ім'ям VISU_1

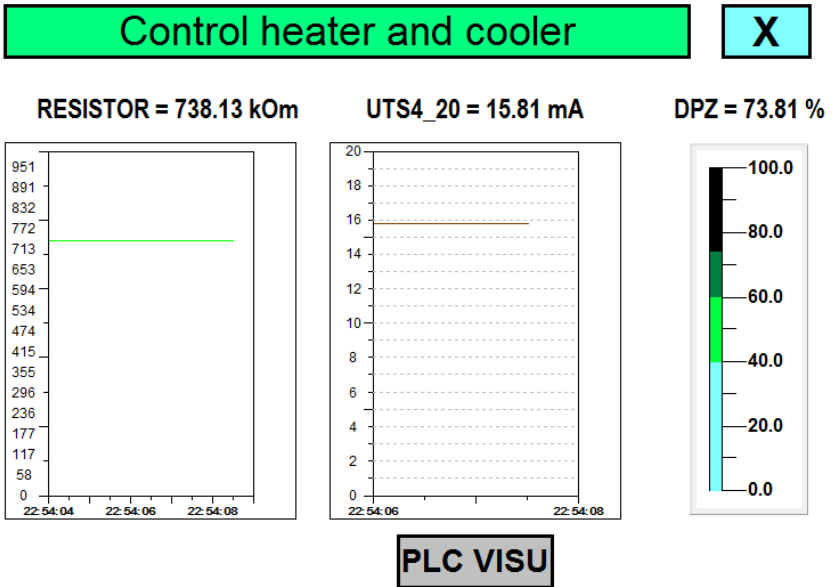


Рис. 1.26 – Приклад візуалізації з ім'ям VISU_2

Збережіть проект та завантажте його до ПЛК. Для запуску програми користувача натисніть на клавіатурі ПК функціональну клавішу F5 або виберіть команду Run в меню Online. За змінними можна спостерігати, якщо відкрити візуалізацію проекту або якщо стежити за значеннями змінних у вікні редактора програми користувача.

1.5. Перевірка роботи навчального проекту

1) Перевірте роботу програми користувача в проекті. Для цього вмикайте перемикачі, змінійте опір резистору та вводіть значення локальних змінних і спостерігайте за всіма змінами в програмі користувача на екрані монітора. Зафіксуйте візуалізацію зробивши декілька скриншотів (за допомогою клавіші PrtSc на клавіатурі ПК). Подайте проект викладачу для перевірки.

2) Складіть звіт в редакторі *Microsoft Office* відповідно до правил оформлення звітів: відомості про виконавця, назву та мету роботи, схему з'єднань, лістинг програми ПЛК, параметри налаштувань COM-порту для завантаження проектів та скриншоти екранів з візуалізаціями в режимі Online. Для документування проекту використайте вбудовану в *CDS2* відповідну утиліту, яка активується натисненням ЛКМ на команду Document в меню Project.

1.6. Завдання для самостійної роботи

1) Розробіть програму користувача, в якій умовою вмикання та вимикання вихідних елементів будуть логічні вирази, які наведені в табл. 1.1. При цьому необхідно скласти таблицю істинності, тобто визначити необхідний стан входів для вмикання виходу. Зробіть візуалізацію проекту.

2) Розробіть програму користувача для дискретного управління тепловим об'єктом: нагрівачем або холодильником. В програмі необхідно реалізувати такі функції:

- вмикання сигналізації при виникненні будь-якої з аварій на об'єкті (використати перемикачі на фізичних дискретних входах);
- вимикання об'єкта при виникненні будь-якої з аварій;
- вмикання об'єкта за допомогою *soft*-кнопки, за умови відсутності аварій;
- вимикання об'єкта за допомогою *soft*-кнопки.

Таблиця 1.1 – Варіанти умов для вмикання вихідних елементів

№	Логічна умова ($X1...X6$ – дискретні входи)	Дискретний вихід (Y)
1	$(X1 \text{ AND } X3) \text{ OR } (X4 \text{ AND } X5 \text{ AND NOT } X6)$	Y3
2	$(X2 \text{ OR } X3) \text{ AND } (\text{NOT } X1 \text{ AND } X5 \text{ AND } X6)$	Y4
3	$(X1 \text{ AND } X3) \text{ OR } X4 \text{ AND NOT } X5 \text{ AND } X6)$	Y3
4	$(X2 \text{ AND } X3) \text{ XOR } (X1 \text{ AND } X5 \text{ AND } X6)$	Y4
5	$(\text{NOT } X1 \text{ XOR } X3) \text{ AND } (X4 \text{ OR } X5 \text{ OR } X6)$	Y3
6	$(X2 \text{ AND NOT } X3) \text{ OR } (X4 \text{ AND } X5 \text{ AND } X6)$	Y4
7	$X2 \text{ AND } X3 \text{ AND NOT } X4 \text{ AND } (X1 \text{ XOR } X6)$	Y3
8	$(X1 \text{ OR } X3 \text{ OR } X4) \text{ AND } (\text{NOT } X5 \text{ OR } X6)$	Y4
9	$(X2 \text{ XOR } X3) \text{ AND } (\text{NOT } X4 \text{ OR } X1 \text{ OR } X6)$	Y3
10	$(X1 \text{ OR } X3 \text{ OR NOT } X4) \text{ AND } X5 \text{ AND } X6$	Y4

Вмикання живлення теплового об'єкта (TEN або VENT) або лампи сигналізації LAMP здійснюється залежно від результату логічної операції (РЛО) над чотирма вхідними дискретними змінними: PUSK, STOP, AVAR та POGAR. Якщо РЛО має статус TRUE, то індикатор об'єкта буде синього кольору. У разі спрацьовування датчиків індикатор LAMP буде червоного кольору, а індикатор об'єкта – сірого. Об'єкт включений до моменту натиснення на кнопку STOP або до спрацьовування датчиків AVAR або POGAR. В табл. 1.2 наведено варіанти використання входів та виходів ПЛК. Зробіть візуалізацію проекту подібно до зображеній рис. 1.27.

Таблиця 1.3 – Варіанти входів та виходів ПЛК

№	PUSK	STOP	AVAR	POGAR	TEN або VENT	LAMP
1	DI1	DI2	DI3	DI4	DO3	DO2
2	DI2	DI3	DI4	DI5	DO4	DO1
3	DI3	DI4	DI5	DI6	DO3	DO2
4	DI4	DI5	DI6	DI1	DO4	DO1
5	DI5	DI6	DI1	DI2	DO3	DO2

3) Розробіть програму користувача для дискретного керування об'єктом типу «ємність для накопичення та зберігання», яка реалізує алгоритм дистанційного керування установкою водопостачання. Схема установки водопостачання зображена на рис. 1.28.

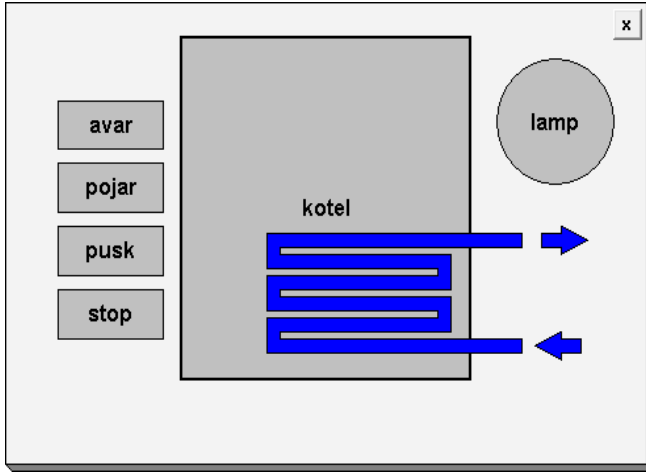


Рис. 1.27 – Приклад візуалізації проекту

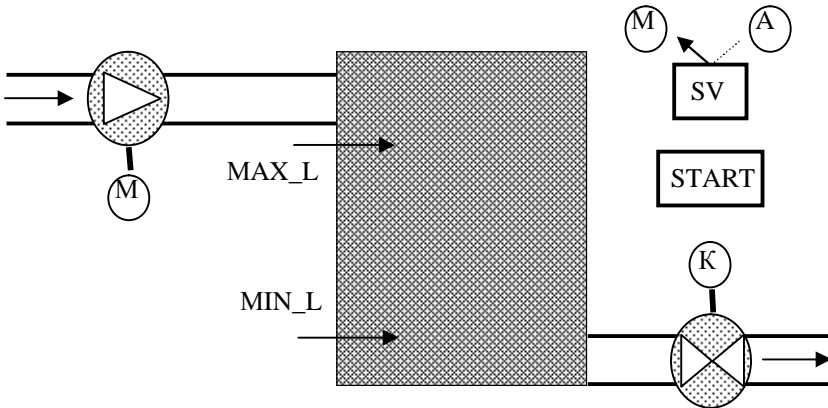


Рис. 1.28. – Схема установки водопостачання

Умови завдання:

Опис системи управління водопостачанням. Об'єктом керування є ємність для накопичення води, яка наповнюється із свердловини за допомогою насоса (*nasos*), який встановлений на вхідному трубопроводі. Вода витрачається для поливу. В ємності встановлено два контакт-

них датчика, які визначають мінімальний (min_l) та максимальний рівні (max_l). Датчики рівня поплавкового типу підключені по схемі *NC*, тобто, нормально-замкнений контакт. Це означає, якщо ємність порожня, то вони у замкненому стані, але, якщо рівень води в ємності поступово збільшується, вони по черзі розмикаються, спочатку датчик мінімального рівня, потім – датчик максимального рівня. В вихідному трубопроводі встановлений електромагнітний клапан (*klapan*). Система управління може працювати у двох режимах: ручному або автоматичному.

Алгоритм роботи системи управління водопостачанням буде таким. Перемикач *SV* визначає режим роботи установки. Якщо його контакти розімкнути, то установка працює в ручному режимі. В протилежному випадку – режим автоматичний.

В ручному режимі, якщо контакти кнопки *START* в розімкненому стані, то установка не працює. В протилежному випадку – установка працює. В цьому режимі враховуються сигнали від датчиків рівня для перемикачів живлення насоса та клапана.

В автоматичному режимі, якщо контакти датчика min_l та max_l замкнені, то ємність порожня. Автоматично вмикається насос та вимикається клапан. Якщо контакти датчика min_l та max_l розімкнуті, то ємність заповнена: автоматично вимикається насос, та вмикається клапан. Якщо контакти датчика max_l розімкнені, контакти датчика min_l замкнені (рівень в ємності середній), то клапан залишається включеним, а насос – не вмикається.

Самостійно проведіть конфігурування ресурсів ПЛК, тобто, виберіть фізичні входи та виходи ПЛК та зв'яжіть їх з відповідними змінними. Також самостійно розробіть візуалізацію установки водопостачання. При цьому використовуйте схему, яка зображена на рис.1.28.

1.7. Контрольні питання

- 1) Наведіть коротку характеристику контролерів ОВЕН.
- 2) Який склад та можливості програмного комплексу *CDS V2*?
- 3) З яких компонентів та ресурсів складається проект в середовищі *CDS V2*?
- 4) Які фізичні інтерфейси використовують для завантаження проекту до ПЛК?
- 5) Яка мета та порядок конфігурування ресурсів ПЛК?
- 6) Поясніть порядок налаштування графічних елементів.

Практичне завдання 2

СТРУКТУРУВАННЯ ПРОЕКТІВ ТА ОСНОВНІ ПРИНЦИПИ ВИКОРИСТАННЯ ТАЙМЕРІВ ТА ЛІЧИЛЬНИКІВ

2.1. Мета завдання

- закріплення теоретичних знань щодо використання програмних організаційних компонентів в проекті та його структурування;
- засвоювання методики розроблення програм користувача з використанням таймерів та лічильників.

2.2. Порядок виконання завдання

Виконання практичного завдання складається з таких етапів:

1) Створення проекту та розроблення програми користувача з *POU* типу PROGRAM (PRG) – для ознайомлення з роботою таймерів та лічильників.

2) Створення проекту та розроблення програми користувача – управління насосом на трубопроводі подавання сировини зі створенням *POU* типу Function Block (FB) користувача.

2.3. Хід виконання завдання

2.3.1. Створення проекту в середовищі CDS2 для ознайомлення з роботою таймерів та лічильників

Проведіть підготовчі роботи відповідно до дій, які описані в практичному завданні №1 (див. пункт 1 в підр. 1.3 та пункти 1.4.1 – 1.4.3). Але при створенні головного *POU* з ім'ям PLC_PRG мову програмування виберіть мову *ST* (текстова мова, подібна до мови *C*).

В менеджері проектів за допомогою контекстного меню послідовно створіть два нових *POU* типу PROGRAM з мовою реалізації *CFC* та ім'ям *counters* та *timers*.

У робочому вікні введіть рядки для виклику *POU counters* та *timers*, а саме для виклику програм з таймерами та лічильниками. В результаті вигляд головного *POU* буде відповідати екранній формі, яка зображена на рис. 2.1. Зауважимо, що *POU* типу PROGRAM викликаються через їхнє ім'я з порожніми скобками.

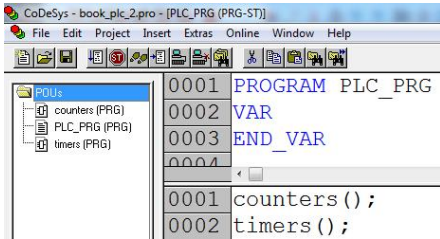


Рис. 2.1 – Вікно головного *POU* з ім'ям *PLC_PRG*

Отже, зараз все готово для подальшого розроблення проекту. Відкрийте для редагування один зі створених *POU* – програму *counters*. До робочого простору програми *counters* послідовно додайте три блочних елементи. За умовчанням вони при додаванні до робочого простору будуть типу *AND*. Виділіть

ЛКМ один із створених елементів і далі використайте асистент вводу для заміни елемента *AND* на елемент *CTD* зі списку стандартних функціональних блоків, які входять до переліку *Counter* в бібліотеці *STANDARD.lib*. Ця бібліотека за умовчанням автоматично підключається до проекту під час його створення. На рис. 2.2 зображена екранна форма, на якій показано вікно вибору елемента декрементного лічильника *CTD (FB)* замість елемента *AND*, який створений раніше.

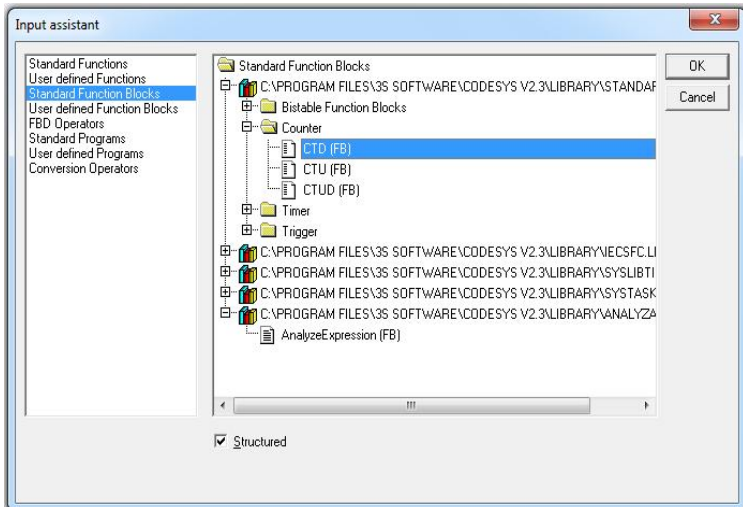


Рис. 2.2 – Вікно вибору декрементного лічильника *CTD*

Подібно до описаних раніше дій замініть два елементи AND, що залишились, на лічильники типу інкрементний (CTU) та інкрементно-декрементний (CTUD). Підключіть до входів та виходів лічильників відповідні входні та вихідні елементи.

Програма `counters` повинна виглядати так, як вона зображена на рис. 2.3. Всі лічильники мають позначені входи та виходи. Пояснення щодо типів даних змінних, які з'єднані з блоками лічильників, можна отримати двома способами. Або за допомогою коментарів в бібліотеці `STANDARD.lib`, або за допомогою вбудованої в `CDS2` довідкової системи. Довідкова система викликається за допомогою функціональної клавіші `F1`.

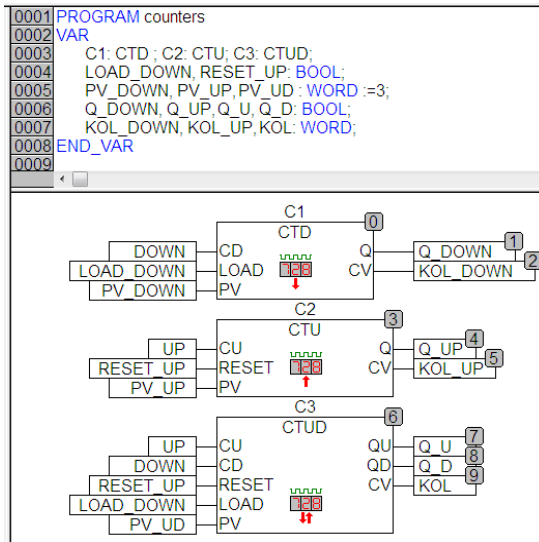


Рис. 2.3 – Вікно програми `counters`

Надамо пояснення щодо рис. 2.3. В зоні оголошення змінних показані екземпляри лічильників (C1: CTD; C2: CTU; C3: CTUD;), входи для керування лічильниками (LOAD_DOWN, RESET_UP: BOOL;), змінні для встановлення параметра лічення (PV_UP, PV_UD, PV_DOWN: WORD :=3;), виходи для відображення стану

лічильників (Q_DOWN, Q_UP, Q_U, Q_D: BOOL;) та змінні для зберігання поточного значення в лічильнику (KOL_DOWN, KOL_UP, KOL: WORD;). Лічильні входи (CU та CD) прив'язані до фізичних входів DI1 та DI2 в ПЛК і мають відповідно імена UP та DOWN.

В процесі розроблення візуалізації програми користувача створіть графічні елементи **Ellipse** для відображення стану фізичних входів ПЛК та стану лічильників. Для входів керування лічильниками використовуйте графічний елемент **Button**. Для відображення та введення параметра лічення використовуйте графічний елемент **Rectangle**. Для відображення поточного значення лічення теж використовуйте графічний елемент **Rectangle**.

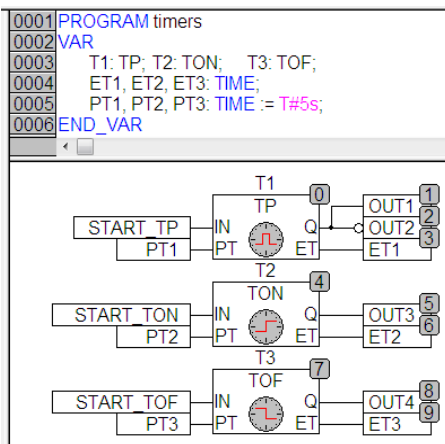


Рис. 2.4 – Вікно програми *timers*

Для запуску таймерів використовуйте в програмі змінні **START_TP**, **START_TON** та **START_TOF**, які зв'язані з фізичними дискретними входами ПЛК DI4, DI5 та DI6. Для усіх таймерів створіть змінні типу **TIME** (PT1, PT2, PT3: TIME := T#5s;) для введення параметру часу до таймерів. Для відображення поточного значення часу використовуйте змінні типу **TIME** (ET1, ET2, ET3). Стан таймерів зв'яжіть з фізичними дискретними виходами ПЛК OUT1, OUT2, OUT3 та OUT4.

Збережіть проект з ім'ям *lr_2_1_name.pro*. Зробіть візуалізацію для одного з програмних компонентів за вказівкою викладача та подайте йому отримані результати. Візуалізація має виглядати подібно до зображеної на рис. 2.5. Налаштуйте зв'язок середовища *CDS2* з ПЛК та завантажте проект до ПЛК. Перевірте правильність роботи програми користувача.

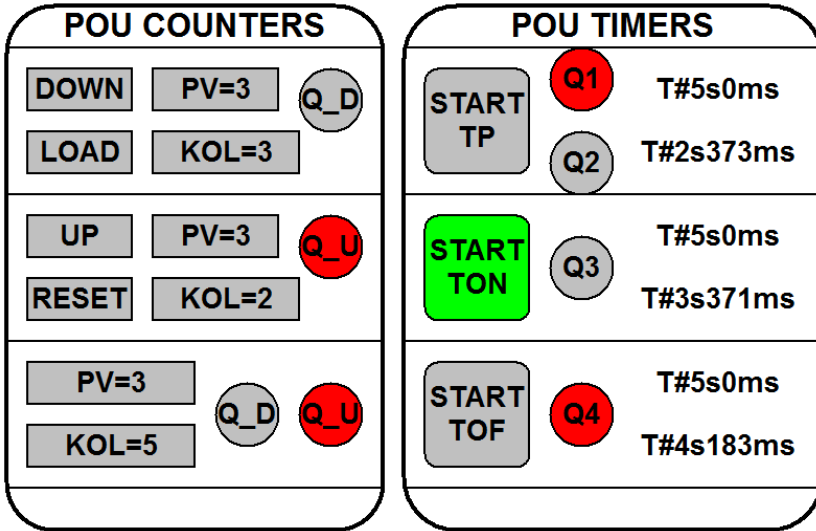


Рис. 2.5 – Візуалізація POE counters та timers

2.3.2. Створення проекту з програмою керування насосами

Розробіть проект з програмою керування двома насосами на трубопроводі подавання сировини. Спочатку розробимо програму керування одним насосом. Потім створимо функціональний блок для керування насосом.

Кнопкою Push вмикається електричний двигун насоса. Насос повинен працювати на протязі 10 секунд. Далі насос автоматично вимикається. Для вмикання насоса потрібно знову натиснути на кнопку Push. Одночасно в програмі потрібно рахувати кількість вмикань насоса. Програма керування насосом зображена на рис. 2.6. Зробіть візуалізацію процесу, подібну до зображеної на рис. 2.7. Далі збережіть проект з ім'ям *lr_2_2_name.pro*.

Налаштуйте зв'язок середовища CDS2 з ПЛК та завантажте проект до ПЛК. Перевірте правильність роботи програми користувача.

В процесі налагодження проекту під час роботи насоса натисніть ЛКМ на кнопку Push. Спостерігайте за значенням змінної кількості

вмикань. Якщо є застереження, що програма працює неправильно (наприклад, хибне значення кількості вмикань), то виправте цю помилку самостійно. Тобто, змініть вміст програми так, щоб не було хибного рахування вмикань насоса.

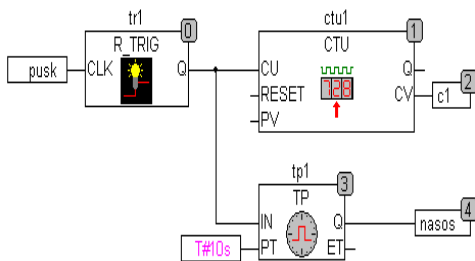


Рис. 2.6 – Програма управління насосом



Рис. 2.7 – Візуалізація програми управління насосом

Модифікуйте проект для керування двома насосами. Для цього створить функціональний блок з алгоритмом керування насосом. В головному *POU PLC_PRG* створить два екземпляри для керування двома насосами. Модифікуйте також візуалізацію, яка зображена на рис. 2.7 для керування двома насосами. Передбачте можливість введення часу роботи насоса та скидання кількості вмикань

2.3.3. Створення ярлика для запуску проекту без використання середовища CoDeSys V2.3

Для запуску проекту з візуалізацією без запуску середовища *CDS2* створить ярлик виклику проекту, як це робиться зазвичай в операційній системі *Windows*. За допомогою контекстного меню відкритте властивості ярлика. У вікні властивостей перейдіть на вкладку *Ярлик*, а в ній заповніть поля *Об'єкт* та *Робоча папка*. Для поля *Об'єкт* рядок для заповнення буде таким: "C:\Program Files\3S Software\CoDeSys V2.3\ CoDeSysHMI\CoDeSysHMI.exe"/target"D:\OWEN\1_nas_vkl.pro". У даному рядку перша частина вказує місце розташування файлу запуску програвача візуалізацій *CoDeSysHMI.exe*, а друга частина після символу */* та параметра *target* вказує місце розміщення проекту на жор-

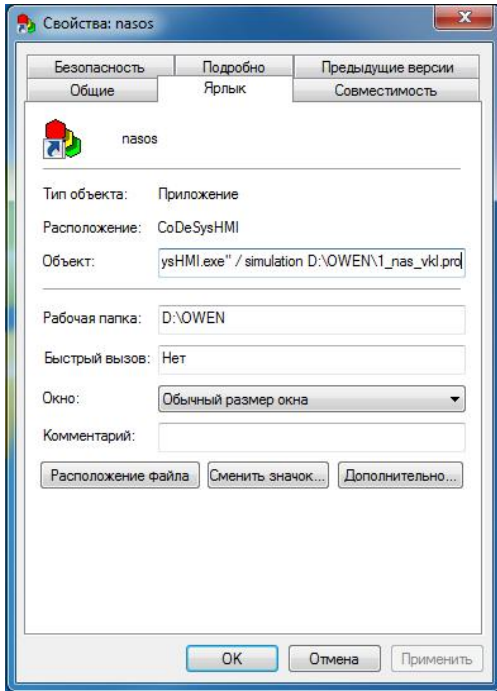


Рис. 2.8 – Вікно налаштувань властивостей ярлика для виклику проекту

сткому диску ПК. Параметр `target` вказує, що проект буде запуснений на реальному ПЛК. Якщо використати параметр `simulation`, тоді проект буде запуснений режимі симуляції. В поле *Рабочая папка* введіть шлях до директорії з проектом: `D:\OWEN`. Екранна форма налаштувань властивостей ярлика зображена на рис. 2.8. Взагалі порядок роботи з програвачем візуалізації можна отримати з довідкової системи, яка додається до середовища *CDS2* (див. розділ *CoDeSys HMI*).

Подайте викладачу проект з можливістю його

виклику за допомогою ярлика з реалізацією на емуляторі або на цільовій платформі.

2.3.4. Створення бібліотеки користувача на прикладі проекту з функціональним блоком

Відкрийте проект з ім'ям `lr_2_2_name.pro`. Для створення власної бібліотеки з метою подальшого багатократного використання розробленого готового функціонального блока дійте так.

По-перше, збережіть проект з новим ім'ям.

По-друге, створіть *POU* типу *Function Block*, перенесіть до нього програму з головного *POU* (див. рис. 2.6), далі видаліть з проекту головний *POU* з ім'ям `PLC_PRG`.

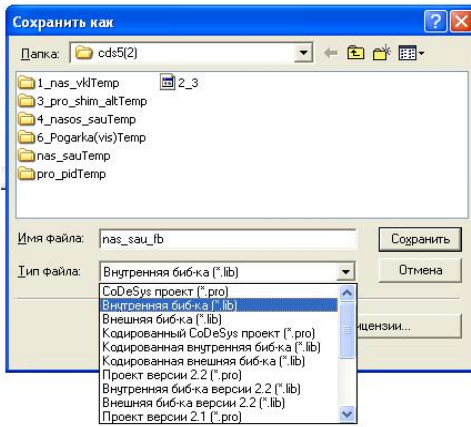


Рис. 2.9 – Вікно для створення бібліотеки користувача

По-третє, в меню File виберіть команду Save as.... Далі у вікні Save as, екранна форма якого зображена на рис. 2.9, дайте назву власному бібліотечному функціональному блоку в полі **Имя файла:** – nasos. У полі **Тип файла:** виберіть з випадаючого меню тип бібліотеки – **Внутренняя биб-ка (*.lib)**. Далі натисніть на кнопку **Сохранить**. В результаті буде скомпільована бібліотека користувача з

функціональним блоком для керування насосом. Докладнішу інформацію про типи бібліотек можна отримати в довідковій системі *CDS2*.

2.4. Перевірка роботи навчального проекту

1) Перевірте роботу програми користувача в проекті. Для цього вмикайте перемикачі та вводіть значення локальних змінних і спостерігайте за всіма змінами в програмі користувача на екрані монітора. Зафіксуйте візуалізацію зробивши декілька скриншотів. Подайте проект викладачу для перевірки.

2) Складіть звіт в редакторі *Microsoft Office* відповідно до правил оформлення звітів: відомості про виконавця, назву та мету роботи, схему з'єднань, лістинг програми ПЛК, параметри налаштувань *COM*-порту для завантаження проектів та скриншоти екранів з візуалізаціями в режимі *Online*. Для документування проекту використайте вбудовану в *CDS2* утиліту *Document* в меню *Project*.

2.5. Завдання для самостійної роботи

1) Удосконалення проекту з програмою керування насосом

Розробіть проект з програмою керування насосом на трубопроводі подавання сировини з наступним алгоритмом: вимірювання часу роботи

насосу та захист роботи насоса від «сухого ходу». Кнопкою **Push** вмикається електричний двигун насоса. Нехай для забезпечення захисту насосу від «сухого ходу» в трубопроводі вбудований датчик тиску **dat**. Якщо на протязі п'яти секунд не спрацює датчик тиску, то насос автоматично вмикається через три секунди. Доопрацюйте проект з програмою керування насосом, як це зображено на рис. 2.10. Модифікуйте візуалізацію, яка зображена на рис. 2.7 так, як це показано на рис. 2.11.

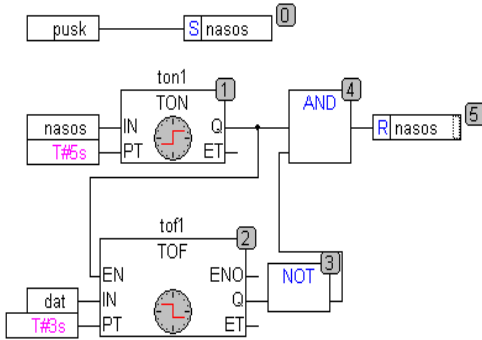


Рис. 2.10 – Програма управління насосом з датчиком тиску

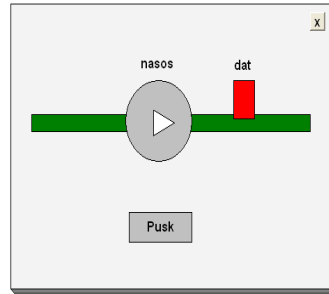


Рис. 2.11 – Візуалізація проекту системи управління насосом

2) Розробіть проект, що реалізує дискретне керування насосом з використанням таймерів, а також датчика тиску аналогової дії.

Схема установки водопостачання подібна до зображеної на рис. 1.23, а умови роботи описані в третьому завданні для самостійної роботи в першому практичному завданні. Відмінність полягає в тому, що замість сигналізаторів рівня використаний аналоговий датчик рівня. Для імітування датчика рівня використайте змінний резистор, а поріг для відключення встановіть на рівні 10 % від всього діапазону (0...1000 од.). Зробіть візуалізацію проекту з додаванням графічного елементу типу **Rectangle** для відображення значення рівня в ємності.

3) Розробіть проект, що реалізує дискретне керування насосом та електроклапаном. Як приклад використайте усі проекти, що запропоновані раніше.

Необхідно додатково реалізувати у проекті наступні функції:

– якщо на протязі п'яти секунд датчик тиску досягне потрібного значення (30 % від всього діапазону), то відкривається випускний електроклапан;

– якщо тиск знизиться до мінімального рівня (10 % від всього діапазону), то випускний електроклапан закривається.

Зробіть візуалізацію проекту, подібну до зображеної на рис. 2.11, але додайте графічний символ для відображення стану електроклапана та значення тиску в трубопроводі.

2.6. Контрольні питання

- 1) Дайте коротку характеристику таймерів згідно з *IEC61131*.
- 2) Дайте коротку характеристику лічильників згідно з *IEC61131*.
- 3) Як використовують таймери та лічильники в програмах керування зовнішніми пристроями?
- 4) Дайте коротку характеристику бібліотеки *Standard.lib*.
- 5) Надайте характеристику *POU* типу функція, функціональній блок та програма.
 - б) У чому полягає різниця між внутрішньою та зовнішньою бібліотеками в середовищі *CDS2*?

Практичне завдання 3
**ПРИНЦИПИ РЕАЛІЗАЦІЇ ДВОХПОЗИЦІЙНОГО
ТА ШІМ-РЕГУЛЮВАННЯ**

3.1. Мета завдання

- закріплення теоретичних знань щодо принципів двохпозиційного управління дискретними вихідними пристроями;
- закріплення теоретичних знань щодо принципів аналогового управління дискретними вихідними пристроями (ШІМ);
- засвоєння порядку налаштування апаратного ШІМ-регулятора на прикладі конфігурування ресурсів ПЛК ОВЕН;
- розроблення програми користувача для ШІМ-регулювання на прикладі використання стандартних функціональних блоків з бібліотеки Util.lib.

3.2. Порядок виконання завдання

Виконання лабораторної роботи складається з таких етапів:

- 1) Створення проекту та розроблення програми двохпозиційного сигналізатора або регулятора параметра.
- 2) Створення проекту та розроблення програми ШІМ-регулятора на прикладі використання імпульсного таймера.
- 3) Створення проекту та розроблення програми ШІМ-регулятора на прикладі використання блока BLINK з бібліотеки Util.lib.

3.3. Хід виконання завдання

3.3.1. Створення проекту с програмою двохпозиційного сигналізатора параметра

Необхідно реалізувати в проекті такі функції:

- вимірювання температури за допомогою датчика, який підключений до аналогового входу;
- включення сигналізації (лампа), якщо температура буде вища від заданого максимального значення (за умовчанням 150°C);
- відключення сигналізації, якщо температура буде нижча від заданого мінімального значення, яке менше за перше (за умовчанням 100°C).

Нехай аналоговий сигнал імітується змінним опором, який підключений до входу AI4 ПЛК (встановить верхню межу АЦП на рівні 1000 од., тоді діапазон зміни параметра буде складати від 0 до 200 °С). Зробіть візуалізацію процесу, подібну до зображеної на рис. 3.1.

Збережіть проект з ім'ям lr_3_1_name.pro. Заповніть *POU* типу Program з ім'ям PLC_PRG згідно з лістингом:

```
PROGRAM PLC_PRG
VAR
    ust1: REAL := 150;
    ust2: REAL := 100;
END_VAR
IF temp>ust1 THEN lamp:=TRUE;
END_IF
IF temp<ust2 THEN lamp:=FALSE;
END_IF .
```

Подайте викладачу проект з можливістю його виклику за допомогою ярлика з реалізацією на цільовій платформі.

3.3.2. Загальні відомості про принципи 2-х позиційного регулювання

Регулятор – це пристрій, який призначений для підтримки контрольованої величини на заданому рівні. Для цього можна використовувати вихідний пристрій з двома станами: включено або виключено. Причому для підтримки заданого значення регульованого параметра можна використовувати різні типи виконавчих пристроїв, але всі вони можуть бути умовно розділені на дві групи: *нагрівачі* та *охолоджувачі*.

Нагрівачем називають пристрій, включення якого повинне при-

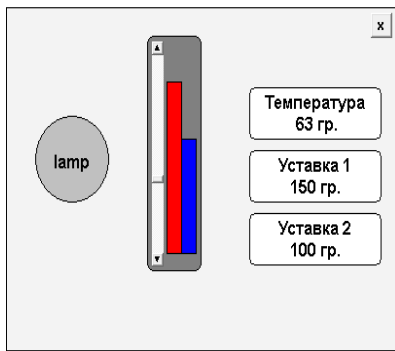


Рис. 3.1 – Візуалізація проекту сигналізатора параметра

водити до збільшення значення вимірюваного параметра. В цьому випадку процес регулювання має прямий характер.

Охолоджувачем називають пристрій, включення якого повинне приводити до зменшення значення вимірюваного параметра. В цьому випадку процес регулювання має зворотний характер.

Алгоритм сигналізатора можна модифікувати, якщо додати до програми гістерезис, тобто допустиме відхилення параметра від завдання. Так замість двох уставок можна використати одну уставку з гістерезисом. Нижче наведено приклад програми терморегулятора в режимі прямого гістерезиса, тобто *нагрівача*:

```
PROGRAM PLC_PRG
```

```
VAR
```

```
    gist: REAL := 5;
```

```
    rele_flag: BOOL;
```

```
END_VAR
```

```
IF (temperatura < (ust-gist) ) THEN (*якщо температура менше завдання*)
```

```
    rele:=TRUE;                (*ввімкнути реле нагрівача*)
```

```
    rele_flag:=TRUE;          (*встановити флаг реле*)
```

```
END_IF;
```

```
IF (temperatura > (ust+gist) ) THEN (*якщо температура вище завдання*)
```

```
    rele:=FALSE;              (*вимкнути реле нагрівача*)
```

```
    rele_flag:=FALSE;        (*скинути флаг реле*)
```

```
END_IF;
```

Змінні в програмі прив'язані до відповідних фізичних входів та виходів ПЛК згідно з електричною схемою, яка зображена на рис. 1.2. Аналогічним чином буде виглядати програма терморегулятора в режимі *охолоджувача*. Але умовою вмикання вихідного пристрою буде $temperatura > ust + gist$, а вимикання – $temperatura < ust - gist$.

Створіть проект з ім'ям *lr_3_2_name.pro* згідно із завданням, розробіть візуалізацію та подайте проект викладачу.

3.3.3. Загальні відомості про принципи ШІМ-регулювання

Сигнал ШІМ – це імпульсний сигнал постійної частоти та змінної шпаруватості (тривалість імпульсів). За допомогою завдання шпаруватості можна змінювати потужність на дискретному виході ПЛК. Наприклад, це дозволяє регулювати температуру в термошафі з дискретними нагрівальними елементами – інфрачервоними лампами.

Шпаруватість – одна з класифікаційних ознак імпульсних систем, яка визначає відношення періоду слідування сигналу до тривалості імпульсу. Величина, яка обернена шпаруватості, в англійській літературі називається коефіцієнтом заповнення (від англ. *Duty Cycle*).

Таким чином, для імпульсного сигналу справедливі такі співвідношення: $S = T / \tau = 1 / D$, де S – шпаруватість, D – коефіцієнт заповнення, T – період слідування імпульсів, τ – тривалість імпульсу.

Найчастіше застосування в практиці знаходить сигнал з шпаруватістю, яка дорівнює двом. Це так званий сигнал типу «меандр».

В режимі аналогового регулювання логічних пристрій розраховує відхилення E (тобто, розузгодження) поточного значення контрольованої величини T від заданого завдання $T_{заб.}$. В результаті на виході регулятора виробляється аналоговий сигнал Y , який направлений на зменшення розузгодження E . Цей сигнал подається на виконавчий пристрій регулятора у вигляді послідовності імпульсів (ШІМ).

Якщо вихідний пристрій регулятора ключового типу (реле, транзисторна або симісторна оптопара, вихід для управління твердотільним реле), вихідний сигнал перетвориться в послідовність управляючих імпульсів з тривалістю D :

$$D = Y \frac{T_{cl}}{100\%},$$

де D – тривалість імпульса, с; T_{cl} – період слідування імпульсів, с (задається користувачем у програмі); Y – вихідний сигнал регулятора.

Принципи формування ШІМ-сигналу для «нагрівача» при різних значеннях вихідного сигналу Y зображені на рис. 3.2. На рис. 3.3 показані діаграми ШІМ-сигналів для різних значень шпаруватості.

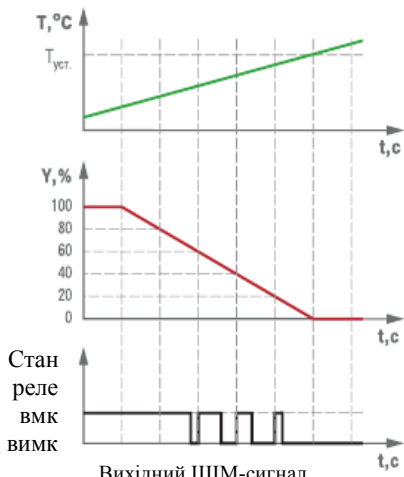


Рис. 3.2 – Принцип формування ШІМ-сигнала

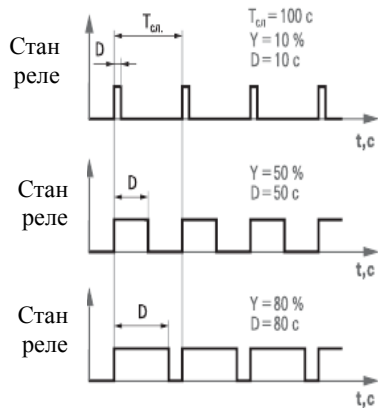


Рис. 3.3 – ШІМ-сигнал для різних значень шпаруватості

3.3.4. Апаратний ШІМ-регулятор

Для закріплення теоретичних відомостей щодо принципів ШІМ-регулювання на прикладі використання апаратного дискретного виходу створіть новий проект з ім'ям *lr_3_3_name.pro*. В якості цільової платформи виберіть *ОВЕН ПЛК150.I-L*. Мову реалізації *POU* з ім'ям *PLC_PRG* оберіть *CFC*.

Для встановлення значення параметра шпаруватості використовуйте змінний резистор, який підключений до аналогового входу ПЛК. Для реалізації апаратного ШІМ-регулятора налаштуйте дискретний вихід ПЛК на роботу в режимі формування ШІМ-сигналу. Для цього перейдіть до вікна конфігурування ПЛК. Відкрийте фіксований слот дискретних виходів. Далі за допомогою ПКМ викличте контекстне меню, а в ньому виберіть команду *Add Pulse-wide modulator*. Таким чином, отримуєте у складі фіксованого слоту дискретних виходів *Discrete output 4 bit [FIX]* додатковий канал *Pulse-wide modulator [VAR]*. На рис. 3.4 зображена екранна форма вікна зі створеним каналом ШІМ.

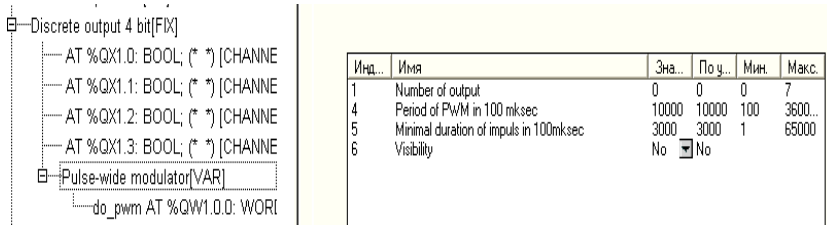


Рис. 3.4 – Вікно налаштування каналу формування ШІМ-сигналу

В параметрах налаштування каналу необхідно вказати номер фізичного виходу, до якого підключений вихідний пристрій. Зауважимо, що нумерація виходів ПЛК починається з нуля. Тобто, якщо нагрівач підключений до виходу *DO4*, то в параметрах налаштувань модуля *Number of output* в першому рядку необхідно ввести число «3». Додатково в параметрах модуля необхідно встановити період слідування імпульсів необхідною кількістю часових інтервалів по 100 мкс. За умовчанням встановлена кількість інтервалів дорівнює 10000. Таким чином, період слідування імпульсів розраховується згідно з наступним виразом: $T=100\text{мкс}\cdot 10000=1\text{с}$. Максимальне можливий період становить 3600000 інтервалів, що відповідає 360 секундам. Аналогічним чином можна встановити мінімальний час тривалості імпульсу. Це потрібно для забезпечення експлуатаційних характеристик вихідного пристрою. Мінімальна кількість циклів складає 100 одиниць (по 100 мкс). В такому випадку період слідування імпульсів буде таким: $T=100\text{мкс}\cdot 100=10\text{мс}$. Це налаштування дозволяє врахувати особливості експлуатаційних характеристик вихідного пристрою, наприклад, теплову інерцію.

Спочатку встановіть період слідування імпульсів на рівні 1 с. Далі за допомогою змінного резистора можна встановлювати час тривалості імпульсу. Але при цьому потрібно врахувати, що значення резистора на аналоговому вході зберігається в змінній типу *REAL* і може набувати значень від 0 до 1000 (якщо в параметрах налаштувань встановити верхню межу на рівні 5000). На виході каналу *Pulse-wide modulator* значення параметра зберігається в змінній типу *WORD* і може набувати

значення від 0 до 65535. Тому в програмі користувача значення резистора потрібно спочатку масштабувати (як це показано в першому практичному завданні, див. пункт 1.4.2), а потім перетворити в змінну типу WORD. Приклад програми користувача зображений на рис. 3.5.

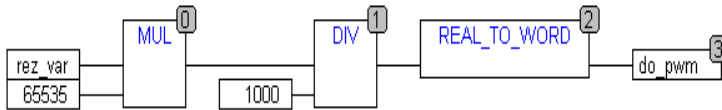


Рис. 3.5 – Програма управління апаратним каналом ШІМ

Таким чином отримаємо проект з можливістю ручного встановлення тривалості імпульсу під час ШІМ-регулювання. Але це лише модель, яка наявно показує, як працює ШІМ-регулятор. Зробіть візуалізацію проекту, в якій створіть тренди для відображення значення вхідного та вихідного параметрів на двох графіках. Подайте проект викладачу для перевірки. Побудуйте графік залежності тривалості імпульсу при ШІМ-регулюванні від значення параметра на вході ПЛК.

3.3.5. ШІМ-регулятор на основі ФБ BLINK з бібліотеки Util.lib

Створіть проект з ім'ям *lr_3_4_name.pro*. В якості цільової платформи виберіть *ОВЕН ПЛК150.1-L*. Мову реалізації *POU PLC_PRG* виберіть *CFC*.

Якщо в ПЛК не передбачена можливість апаратного формування ШІМ-сигналу (наприклад, немає модуля **Pulse-wide modulator**), то цей недолік можна компенсувати за рахунок програмного способу його формування. Але в цьому випадку потрібно одночасно формувати часові інтервали для періоду слідування імпульсів та власне управляти тривалістю імпульсів. Це завдання можна вирішити за допомогою стандартного блоку **BLINK** з бібліотеки *Util.lib*. Цей блок формує прямокутні імпульси визначеної тривалості та періоду, тобто є так званим мульти-вібратором.

Якщо при створенні проекту бібліотека *Util.lib* не була підключена, зробіть це зараз. Для цього за допомогою менеджера проекту перейдіть до ресурсів ПЛК. Подвійним кліком ЛКМ по рядку **Library Manager** відк-

рийте вікно управління бібліотечними ресурсами. З'явиться вікно менеджера бібліотечних ресурсів, екранна форма якого показана на рис. 3.6.

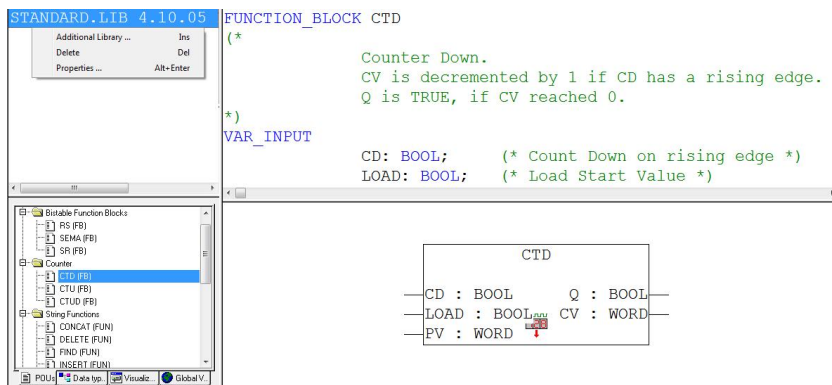


Рис. 3.6 – Вікно менеджера бібліотечних ресурсів

На рис. 3.6 вікно менеджера розділене на декілька зон: зона ліворуч зверху – для управління бібліотеками, зона ліворуч знизу – зі структурою відміченої бібліотеки, поле праворуч зверху – зі змінними активованого елемента бібліотеки та поле праворуч знизу – з позначенням вибраного елемента бібліотеки. У вікні для управління бібліотеками за допомогою ПКМ активовано контекстне меню з командами: **Additional library...**, **Delete** та **Properties...**. Зауважимо, що покажчик миші при цьому повинен знаходитися в полі з переліком раніше підключених бібліотек, тобто зверху ліворуч. Отже, наступним кроком виберіть команду **Additional library...**. Далі з'явиться вікно *Открьть* з каталогом **Library**, який знаходиться в каталозі зі шляхом доступу **C:\Program Files\3S Software\CoDeSys V2.3*.***. Відмітьте потрібну бібліотеку (**Util.lib**) та натисніть ЛКМ на кнопку *Открьть*. Взагалі місце знаходження бібліотек можна встановити або змінити, якщо використати в меню **Project** команду **Options...**, а у вікні налаштувань опцій вибрати категорію **Directories**.

Як в попередніх прикладах, використайте змінний резистор для формування часових інтервалів. Вихідний канал використайте з набору

дискретних виходів в конфігурації ПЛК, наприклад DO4, до якого підключений нагрівач. Поточне значення змінної *rez* на четвертому аналоговому вході ПЛК має тип **REAL**. Але для визначення часу імпульсу повинна бути використана змінна типу **TIME**. Тому в програмі потрібно передбачити перетворення значення змінної з типу **REAL** на тип **TIME**. Нехай період слідування імпульсів буде дорівнювати 1000 мс. Тому час імпульсу буде відповідати перетвореному значенню резистора, а час паузи буде дорівнювати значенню $(1000 \text{ мс} - rez)$. Також в програмі можна використати обмеження мінімального та максимального часу імпульсу (оператор **LIMIT**). Заповніть робоче поле *POU* з ім'ям **PLC_PRG** згідно з екранною формою, як це зображено на рис. 3.7.

В результаті розроблено проект з можливістю ручного встановлення тривалості імпульсу в процесі ШІМ-регулювання. Зробіть візуалізацію проекту, в якій створіть тренди для відображення значення вхідного та вихідного параметра на графіках та значення тривалості імпульсу. Подайте проект викладачу для перевірки. Побудуйте графік залежності тривалості імпульсу при ШІМ-регулюванні на дискретному виході від значення параметра на аналоговому вході ПЛК.

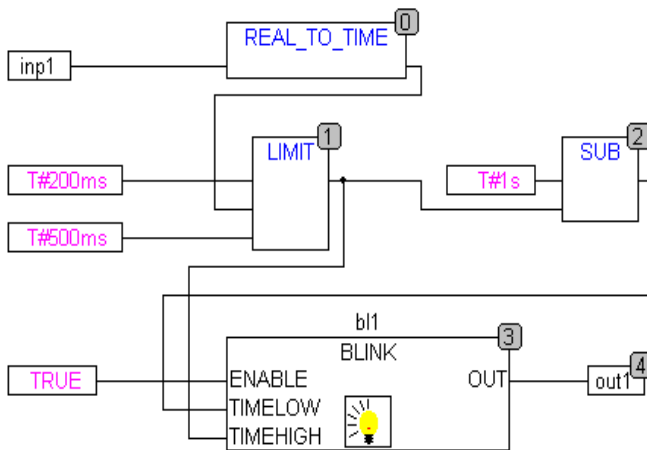


Рис. 3.7 – Програма ШІМ управління вихідним каналом

3.3.6. Багатозадачний режим з ШІМ-регулятором на основі ФБ BLINK

Створіть проект з ім'ям *lr_3_5_name.pro*. В якості цільової платформи виберіть *ОВЕН ПЛК150.I-L*. Створіть три *POU* типу Program з мовою програмування *ST* для розрахунку параметрів ШІМ-регулятора з ім'ям *PLC_PRG*, для реалізації ШІМ-регулювання з ім'ям *PWM_PRG* та для встановлення потрібного значення температури з ім'ям *CONTROL*. Для кожного *POU* необхідно налаштувати задачі за допомогою утиліти Task configuration, яка знаходиться у вкладенні Resources. Встановить покажчик миші в дерево задач Task configuration та за допомогою ПКМ викличте контекстне меню. В меню виберіть команду Insert task. Надайте ім'я для задачі, наприклад, *RECALC*, та прив'яжіть до неї відповідний *POU* (*PLC_PRG*). Для цього ЛКМ виберіть задачу, потім ПКМ викличте контекстне меню і натисніть ЛКМ на команду Append Program Call. У вкладенні Program Call натисніть ЛКМ на кнопку ... та виберіть потрібну програму. Підтвердіть свій вибір натисненням ЛКМ на кнопку ОК. Ця задача буде мати тип суцільс та буде запускати програму *PLC_PRG* з періодичністю 10 с. Подібно до попередніх дій створить другу задачу з ім'ям *PWM_TASK*, прив'яжіть до неї *POU* *PWM_PRG* та встановить час сканування 10 мс. Задача *CONTROL*, яка змінює завдання буде виконуватись з часом сканування 500 мс. В результаті отримаємо дерево з конфігурацією задач, зображення якого наведено на рис. 3.8.

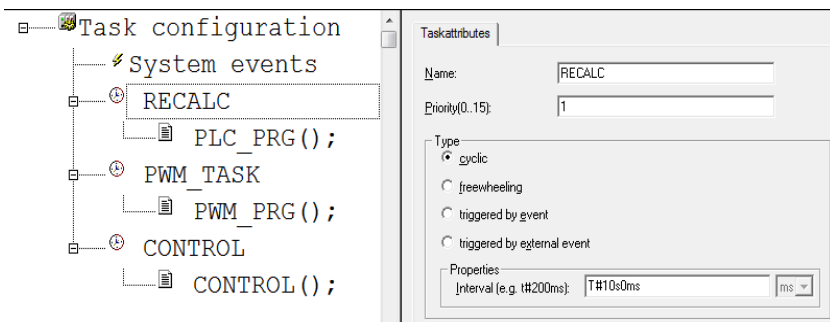


Рис. 3.8 – Дерево конфігурації задач

Заповніть робочий простір створених *POU* відповідно до листингів, які подані нижче.

Лістинг програми PLC_PRG:

```
PROGRAM PLC_PRG
VAR
END_VAR

ACTUAL:=TEMPERATURA;
KD:=100*EXP(-ABS(IDEAL-ACTUAL)/1);
DERRIV:=(ACTUAL-LAST)/REGT;
IF T<IDEAL THEN
    PULSE:=PERIOD*(1-KD*DERRIV);
ELSE
    PULSE:=PERIOD*(-KD*DERRIV);
END_IF;
HEATING:=TRUE;
IF (PULSE<0 OR PULSE=0) THEN
    HEATING:=FALSE;
    PULSE:=0;
END_IF;
IF PULSE>PERIOD THEN
    PULSE:=PERIOD;
    HEATING:=TRUE;
END_IF;
PAUSE:=PERIOD-PULSE;
LAST:=ACTUAL;
```

Лістинг програми PWM_PRG:

```
PROGRAM PWM_PRG
VAR
END_VAR

PWMGEN(ENABLE:=HEATING ,
```

```
TIMELOW:=REAL_TO_TIME(PERIOD*PAUSE*1000),  
TIMEHIGH:=REAL_TO_TIME(PERIOD*PULSE*1000),  
OUT=>PWMOUT);
```

```
OUT:=PWMOUT AND PWMGEN.ENABLE;
```

Лістинг програми CONTROL:

```
PROGRAM CONTROL  
VAR  
END_VAR
```

```
IDEAL:=IDEAL+BOOL_TO_REAL(UP)-BOOL_TO_REAL(DOWN);
```

Перелік змінних, які мають глобальний тип, має наступний склад:

```
VAR_GLOBAL  
(*FOR CONTROL*)  
    UP:BOOL;  
    DOWN:BOOL;  
(*FOR EVAL*)  
    ACTUAL:REAL;(*actual temperature*)  
    LAST:REAL:=35;(*last measured temperature*)  
    IDEAL:REAL:=35;(*temperature must to set*)  
    HEATING:BOOL:=0;  
    KD:REAL;  
    DERRIV:REAL;  
(*CONSTANTS*)  
    PERIOD:REAL:=1; (*period of PWM sec *)  
    REGT:REAL:=10; (*regulation period sec*)  
(*blinker's outs*)  
    PWMOUT:BOOL;      (*control of out*)  
(*blinker*)  
    PWMGEN:BLINK;(*PWM-generator from library "Util.lib"*)  
(*PWM parameters*)  
    PULSE:REAL;  
    PAUSE:REAL;  
END_VAR
```

Поточне значення параметра TEMPERATURA зчитується з 2-го або 3-го аналогових входів ПЛК. Для керування четвертим дискретним виходом ПЛК, до якого підключений нагрівач, використана змінна OUT.

На рис. 3.9 зображена візуалізація, на який відображені тренди та кнопки для зміни завдання.

3.4. Перевірка роботи навчального проекту

1) Перевірте роботу програми користувача в проекті. Для цього вмикайте перемикачі та вводите значення локальних змінних і спостерігайте за всіма змінами в програмі користувача на екрані монітора. Зафіксуйте візуалізацію зробивши декілька скриншотів. Подайте проект викладачу для перевірки.

2) Складіть звіт в редакторі *Microsoft Office* відповідно до правил оформлення звітів: відомості про виконавця, назву та мету роботи, схему з'єднань, лістинг програми ПЛК, параметри налаштувань COM-порту для завантаження проектів та скриншоти екранів з візуалізаціями в режимі Online. Для документування проекту використовуйте вбудовану в CDS2 утиліту Document в меню Project.

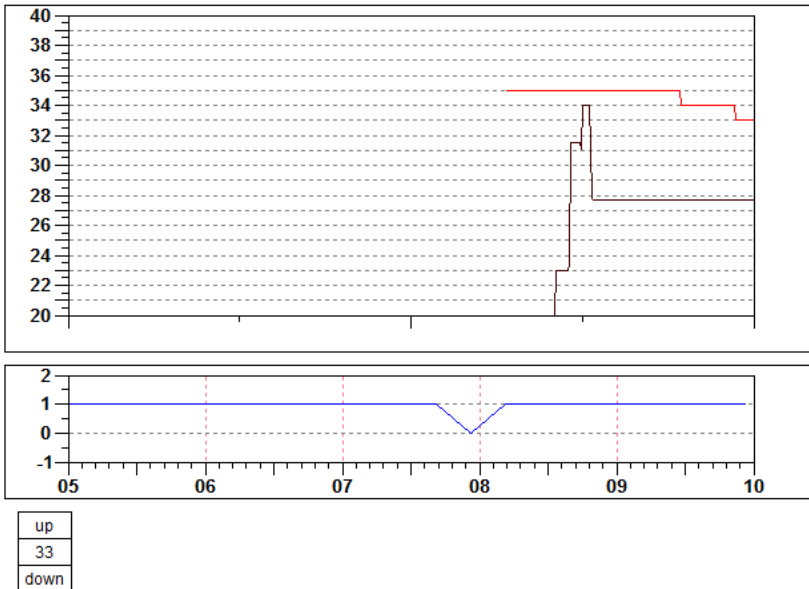


Рис. 3.9 – Візуалізація проекту з ШІМ-регулятором

3.5. Завдання для самостійної роботи

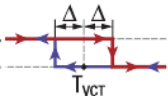
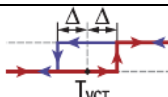
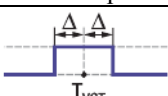
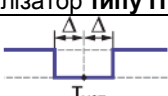
1) Розробіть проект двохпозиційного регулятора з використанням ФБ **HYSTERESIS** з бібліотеки **Util.lib**. Нехай верхня межа буде дорівнювати $35\text{ }^{\circ}\text{C}$, а нижня – $30\text{ }^{\circ}\text{C}$. Створить візуалізацію з можливістю зміни меж.

2) Розробіть проект сигналізатора з використанням ФБ **LIMITALARM** з бібліотеки **Util.lib**. Нехай верхня межа буде дорівнювати $35\text{ }^{\circ}\text{C}$, а нижня – $30\text{ }^{\circ}\text{C}$. Створить візуалізацію з можливістю зміни меж.

3) Розробіть проект, в якому реалізоване дискретне керування нагрівачем або охолоджувачем залежно від поточного значення температури.

Необхідно реалізувати в проекті функції згідно з табл. 3.1. Логіка роботи регулятора вибирається відповідно до номера варіанта. Додатково в проекті передбачте можливість встановлення гістерезису (5 % від всього діапазону) та можливість затримки перемикання вихідного пристрою на три секунди на випадок збурень та похибок при вимірюванні параметра. Зробіть візуалізацію проекту.

Таблиця 3.1 Варіанти умов для вмикання вихідних елементів

№	Тип логіки регулятора	<i>LAMP _TEN</i>	<i>LAMP _VENT</i>	<i>TEN</i>	<i>VENT</i>
1	 «нагрівач»	<i>DO1</i>	-	+	-
2	 «охолоджувач»	-	<i>DO2</i>	-	+
№	Тип логіки сигналізатора	<i>S_OUT</i>		<i>S_IN</i>	
3	 сигналізатор типу П	<i>DO2</i>		<i>DO1</i>	
4	 сигналізатор типу У	<i>DO1</i>		<i>DO2</i>	

В якості датчика виберіть змінний опір, який імітує реальний термометр опору. В програмі користувача має бути реалізоване реальне управління вихідними пристроями (нагрівачем – TЕН або охолоджувачем – VЕНТ). Також повинна здійснюватися світлова сигналізація, що показує стан вихідних пристроїв (LAMP_TЕН і LAMP_VЕНТ) або вхід/вихід контрольованого параметра за вказані межі (S_IN або S_OUT).

В табл. 3.1 рисунки відповідають таким типам регуляторів:

- «нагрівач», тобто 2-позиційний регулятор «прямий гістерезис»;
- «охолоджувач», тобто 2-позиційний регулятор «зворотний гістерезис»;
- П-подібний сигналізатор, тобто вихід спрацьовує при вході параметра в межі;
- У-подібний сигналізатор, тобто вихід спрацьовує при виході параметра за межі.

На рисунках, наведених в табл. 3.1 прийняті такі позначення: $T_{уст}$ – значення уставки (температури), Δ – значення гістерезису.

3.6. Контрольні питання

- 1) Поясніть принцип дії 2-позиційного регулятора параметра.
- 2) Наведіть загальну схему роботи регулятора параметра.
- 3) Дайте коротку характеристику принципу дії ШІМ-регулятора.
- 4) Чим відрізняється апаратний ШІМ-регулятор від програмного?
- 5) Як визначається параметр шпаруватості в апаратному та в програмному ШІМ-регуляторі?
- 6) Яким чином налаштовується багатозадачний режим виконання програм користувача?
- 7) Як визначається порядок виконання задач в багатозадачному проєкті?
- 8) Якого типу задачі можна створити в середовищі CDS2?

Практичне завдання 4

ПРИНЦИПИ РЕАЛІЗАЦІЇ ПІД-РЕГУЛЮВАННЯ

4.1. Мета завдання

- закріплення теоретичних знань щодо принципів реалізації ПІД-регулювання;
- розроблення програм користувача для ПІД-регулювання на прикладі використання стандартних функціональних блоків з бібліотек UTIL.lib та PID_Regulators.lib.

4.2. Порядок виконання завдання

Виконання завдання складається з наступних етапів:

- 1) Створення проекту для ПІД-регулювання 2-позиційним виконавчим пристроєм з використанням ШІМ-управління дискретним виходом.
- 2) Створення проекту ПІД-регулювання 2-позиційним виконавчим пристроєм з автоналаштуванням до параметрів об'єкта регулювання.
- 3) Створення проекту ПІД-регулювання 3-позиційним виконавчим механізмом з регулювальним органом типу «клапан» та автоналаштуванням до параметрів об'єкта регулювання.

4.3. Хід виконання завдання

4.3.1. Короткі відомості про ПІД-регулювання

Здійснення ПІД-регулювання при управлінні технологічними процесами в середовищі CDS2 реалізується за допомогою додаткових бібліотек. Це, наприклад, бібліотека UTIL.lib з папкою Controller, яка входить до інсталяційного пакета CDS2. До цієї папки входять три ФБ: ПІД-, ПІД- регулятори та регулятор з фіксованим часом циклу. Крім того, розробники ПЛК на платформі CDS2 пропонують власні бібліотеки для своїх ПЛК. Так, компанія ОВЕН разом з стандартним набором програмного забезпечення надає додаткові безкоштовні бібліотеки для реалізації ПІД-регулювання та управління різними виконавчими механізмами: PID_Regulators.lib. Але особливістю цих бібліотек є неможливість виконання ФБ бібліотеки в емуляторі ПЛК, тобто для налагодження програм користувача потрібна обов'язкова наявність реального ПЛК.

Крім того, ці бібліотеки не працюють в ПЛК інших виробників. Нагадаємо, що основне завдання ФБ з цих бібліотек – це підтримання поточного параметра (PV) на заданому рівні (SP) шляхом впливу на виконавчі механізми. Загальна схема ПІД-регулятора зображена на рис. 4.1.

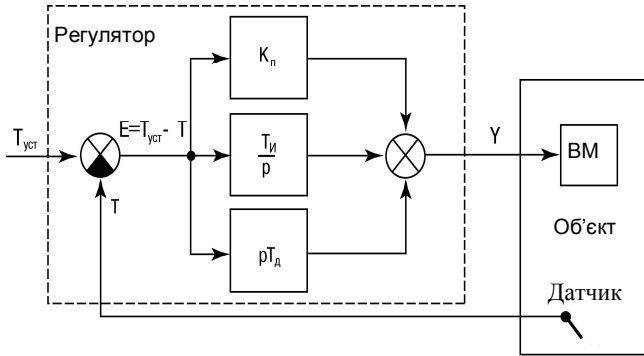


Рис. 4.1 – Загальна схема ПІД-регулятора

Пропорційно-інтегрально-диференційний (ПІД) регулятор використовується в системах автоматики для підтримки з високою точністю потрібних параметрів. Він видає вихідний сигнал, направлений на зменшення відхилення поточного значення регульованої величини від завдання. У загальному випадку робота універсального ПІД-регулятора для вихідного сигналу (Y_i) може бути описана рівнянням

$$Y_i = \frac{1}{X_p} \cdot \left[E_i + \tau_\delta \frac{\Delta E_i}{\Delta t_{вим}} + \frac{1}{\tau_i} \sum_{i=0}^n E_i \Delta t_{вим} \right],$$

де X_p – смуга пропорційності; E_i – різниця між уставкою і поточним значенням контролюваної величини або розузгодження; ΔE_i – різниця між двома послідовними вимірами E_i та E_{i-1} ; $\Delta t_{вим}$ – час між двома послідовними вимірами; τ_δ – постійна диференціювання; τ_i – постійна інтегрування;

випадки; $\sum_{i=0}^n E_i$ – накопичене сумарне розузгодження.

4.3.2. Використання ФБ PID з бібліотеки UTIL.lib для ПІД-регулювання

Для автоматичної підтримки параметру на заданому рівні (наприклад, температури, тиску тощо), можуть використовуватися будь-які модифікації контролерів, що мають аналоговий вхід для вимірювання параметру та дискретний вихід для ШІМ-управління 2-позиційним виконавчим пристроєм або аналоговий вихід для управління за допомогою електричного пневматичного перетворювача мембранним виконавчим механізмом. Розглянемо перший варіант: аналоговий вхід/дискретний вихід.

Завдання для програмування: у сушильній шафі необхідно підтримувати певну температуру. Вибір потрібного значення температури (за умовчанням +40 °С) та перемикання режиму проводиться оператором.

Отже, створіть проект з ім'ям *lr_4_1_name.pro* та підключить до нього бібліотеку UTIL.lib. В якості цільової платформи виберіть *ОВЕН ПЛК150.I-L*. Мову реалізації *POU PLC_PRG* оберіть *CFC*. На рис. 4.2 наведено електричну схему системи регулювання температури. На схемі показано термопара та нагрівач, які підключені до аналогового входу та дискретного виходу відповідно.

Проведіть конфігурування ресурсів ПЛК відповідно до схеми системи регулювання (див. рис. 4.2). При цьому необхідно передбачити можли-

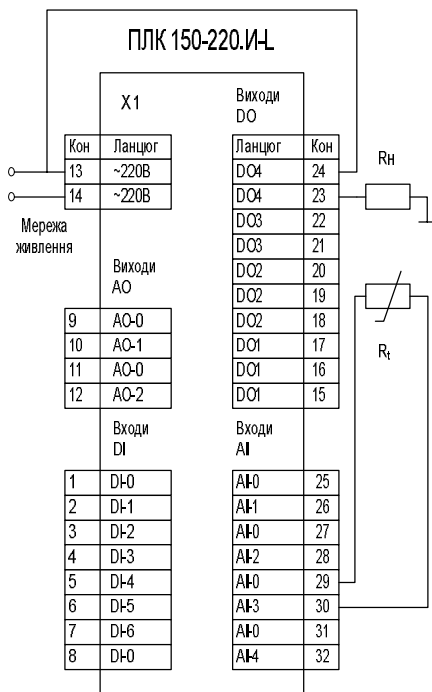


Рис. 4.2 – Схема електричних з'єднань ПЛК

вість вибору потрібного значення уставки за допомогою програмних кнопок UP або DOWN. Нижче наведений перелік змінних, які відносяться до головного *POU* з ім'ям *PLC_PRG*:

```
PROGRAM PLC_PRG
VAR
  BLINK_1: BLINK;
  r_t_1: R_TRIG;
  pd_1: PD;
  ustavka: REAL :=40; (*завдання*)
  up, down: BOOL;
  man, reset: BOOL;
  K_P: REAL := 0.3; (*коефіцієнт пропорційності*)
  Y_out: REAL; (*керуючий сигнал*)
END_VAR
```

На рис. 4.3 зображений фрагмент програми користувача з ПІД-регулятором в режимі «нагрівача». Надамо пояснення до програми користувача. Перший ланцюг, який складають оператори від нульового до третього, в залежності від стану змінних UP або DOWN формує значення уставки. Ці змінні у візуалізації будуть зв'язані з графічними елементами типу Button. Екземпляр функціонального блоку PD з ім'ям *pd_1* виконує розрахунок керуючої дії згідно з пропорційно-диференціальним законом регулювання. В даному блоці не враховується диференціальна складова, тобто закон регулювання є лише пропорційним. Усі вхідні змінні мають інтуїтивно зрозуміле призначення, тому обмежимося поясненням щодо п'ятого та шостого блоків. Ці блоки призначені для перезапуску функціонального блоку *pd_1* кожні 500 мс. В такому випадку в процесі розрахунку керуючої дії не буде накопичуватись інтегральна похибка та буде зменшений негативний ефект від перерегулювання.

Нагадаємо, що блоки з 9-го до 12-го масштабують керуючу дію для реалізації ШІМ-регулювання на дискретному виході ПЛК, до якого підключений нагрівач. При цьому спочатку відкидаються негативні значення сигналу керування, потім здійснюється його масштабування з метою

лінійного приведення сигналу діапазону (0...100 %) до діапазону ШІМ-регулятора (0...65535 од.) і, насамкінець, перетворення змінної типу REAL на тип WORD.

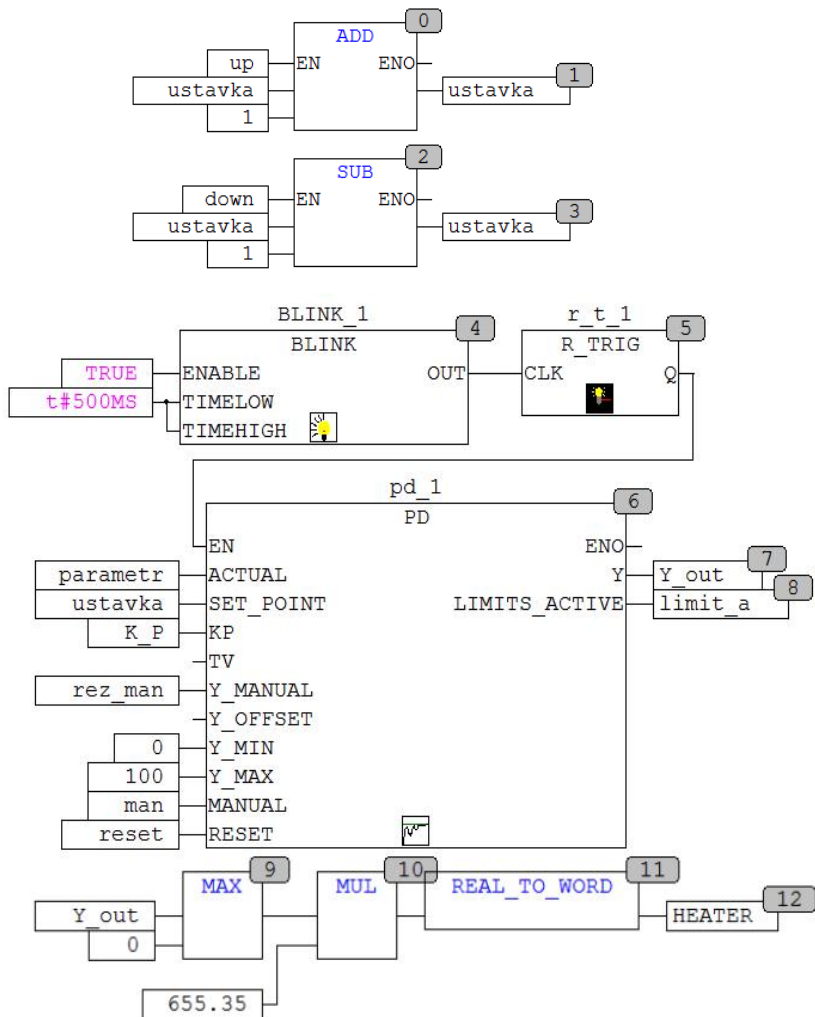


Рис. 4.3 – Фрагмент програми користувача

Аналогічним чином може бути розроблений проект з ПІД-регулятором в режимі «охолоджувача». Керуючий сигнал теж необхідно перетворити, але відкинувши його позитивні значення.

Розробіть візуалізацію проекту з відображенням трендів з поточним параметром, значенням уставки та потужністю управляючого сигналу. В ній також створіть графічний елемент типу **Rectangle** для відображення завдання та кнопки для його змінення. Додатково створіть кнопки **man** та **reset** для керування режимом роботи функціонального блоку **pd_1**. Для ручного встановлення керуючої дії використайте змінну **rez_man**, яка зв'язана з четвертим аналоговим входом. Верхню межу АЦП в налаштуваннях каналу встановіть рівною **500 од**. Це призведе перетворення значення змінного опору до діапазону значень, від **0** до **100 %**. Подайте проект викладачу для перевірки.

4.3.3. Порядок використання ФБ з бібліотеки PID_Regulators.lib

Наступне завдання аналогічне попередньому, але буде використаний ФБ для ПІД-регулювання 2-позиційним механізмом («нагрівачем» або «охолоджувачем») з автоналаштуванням параметрів регулятора по відношенню до об'єкта регулювання. Тому знов використайте схему, що зображена на рис. 4.2. Далі створіть проект з ім'ям *lr_4_2_name.pro* та підключить бібліотеку *PID_Regulators.lib* до нього. В якості цільової платформи виберіть *ОВЕН ПЛК150.1-L*. Мову реалізації *POU* типу програма з ім'ям **PLC_PRG** оберіть *CFC*.

Нижче наведений перелік змінних, які відносяться до головного *POU* з ім'ям **PLC_PRG**:

```
ROGRAM PLC_PRG
VAR
  PID_1: PID_2POS_IM_ANR;
  USTAVKA:REAL :=40;
  Y_out: REAL;
  ANR_START:BOOL;
  ANR_STATE: BYTE;
  df_1: DIG_FLTR;
  P_B: WORD := 2; (*полоса фільтра, в од. пар.*)
  T_I: WORD := 300; (*час фільтра, в мс*)
END_VAR
```

На рис. 4.4 зображений фрагмент програми користувача для ПІД-регулювання за допомогою 2-позиційного виконавчого пристрою та можливістю вмикання та вимикання режиму автоналаштування ПІД-регулятора.

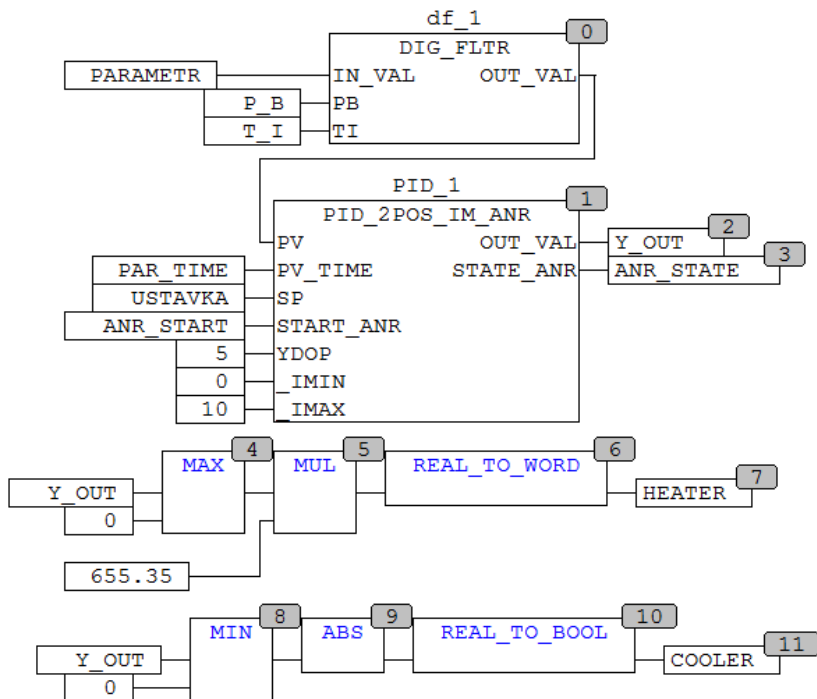


Рис. 4.4 – Фрагмент програми користувача

Зауважимо, що лише змінні **PARAMETR**, **HEATER** та **COOLER** є глобальними та прив'язані до аналогового входу та дискретних виходів ПЛК. Створіть візуалізацію, яка буде відповідати функціональності програмі користувача. В ній передбачте графічні елементи для відображення параметрів та керування ПІД-регулятором. Подайте проект викладачу для перевірки.

Розробіть візуалізацію проекту з відображенням трендів з поточним параметром та значенням розрахованого керуючого сигналу.

4.3.4. Створення проекту з ПІД-регулятором та блоком керування засувкою з датчиком положення із бібліотеки *PID_Rerulators.lib*

На відміну від перших двох проектів, де реалізовано ПІД-регулювання 2-позиційним виконавчим пристроєм (нагрівачем або охолоджувачем), буде розглянута можливість ПІД-регулювання електродвигуном 2-позиційного виконавчого пристрою з регульовальним органом типу «засувка» та автоналаштуванням до властивостей об'єкта.

На рис. 4.5 зображена схема електричних з'єднань ПЛК з зовнішніми пристроями. Керування електричним двигуном здійснюється двома сигналами **MORE** або **LESS**, які забезпечують обертання двигуна в обох напрямках. Відповідно змінюється положення засувки, яка механічно пов'язана зі штоком на валу двигуна. Така схема працює, де використовують механізми електричні прямоходні або однообертові, тобто,

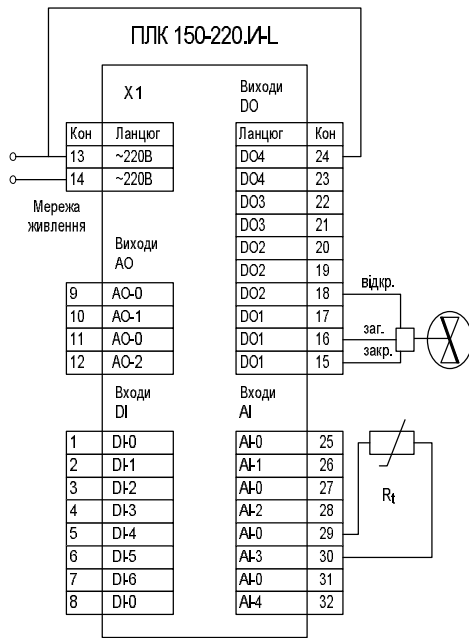


Рис. 4.5 – Схема електричних з'єднань ПЛК з зовнішніми пристроями

МЕП та МЕО. Обертання в різні боки забезпечується комутацією живлення відповідних обмоток. Тому для керування використовують два дискретних виходи ПЛК, причому сигнальні лінії підключені до нормально-відкритих контактів реле.

Програма користувача подібна до програми з попереднього пункту. Але замість формування сигналів для ШІМ-регулювання використовується функціональний блок **VALVE_REG_NO_POS** з бібліотеки *PID_Rerulators.lib*. Тому перелік змінних, які відносяться до головного *POU* з ім'ям **PLC_PRG** буде подібним до переліку, який наведений на стор. 74. Цей перелік буде лише доповне-

ний рядком для об'явлення екземпляра функціонального блоку VALVE_REG_NO_POS з ім'ям VALVE_REG_1. Зауважимо, що в блоці використані константи, які визначають експлуатаційні характеристики виконавчого механізму типу *MEO-86*. Програма користувача зображена на рис. 4.6.

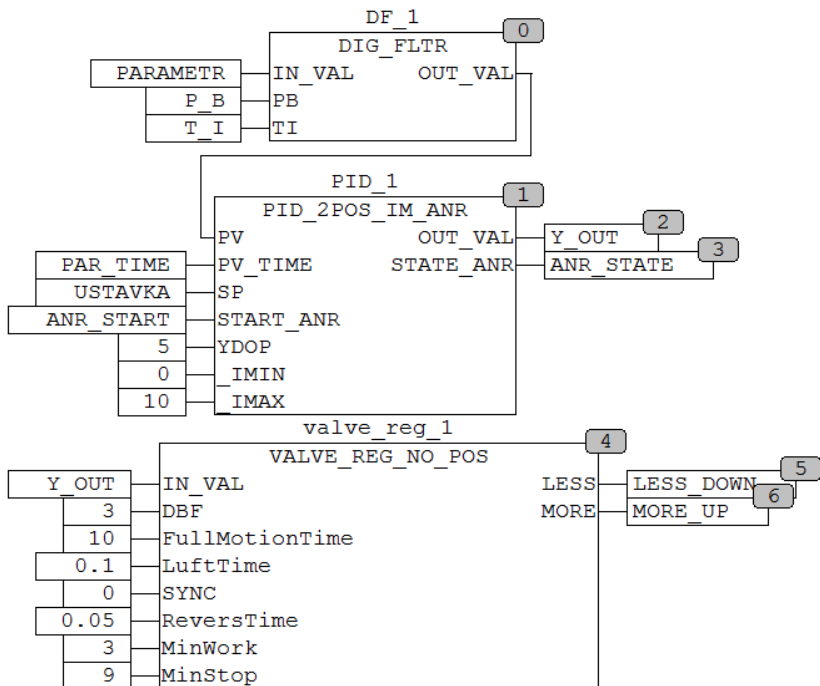


Рис. 4.6 – Програма користувача

Отже, створіть проект з ім'ям *lr_4_3_name.pro* та підключіть бібліотеку *PID_Regulators.lib* до нього. В якості цільової платформи виберіть *ОВЕН ПЛК150.I-L*. Мову реалізації головного *POU PLC_PRG* виберіть *CFC*. При конфігуруванні ресурсів ПЛК використовуйте схему, яка зображена на рис. 4.5.

Розробіть візуалізацію проекту з відображенням трендів з поточним параметром та значенням розрахованої потужності керуючого параметра. Також створіть індикатори режимів керування виконавчим механізмом *MORE* або *LESS*. Подайте проект викладачу для перевірки.

4.3.5. Налаштування конфігуратора тривог

В середовищі *CDS2* є можливість налаштування конфігуратора тривог. Це сигнальна система, яка вбудована в *CDS2*, дозволяє виявляти критичні стани процесу, записувати та візуалізувати їх для користувача за допомогою елементів візуалізації. Механізм роботи сигнальної системи може виконуватися в середовищі *CDS2* або альтернативно в ПЛК. Оброблення тривог в ПЛК задається опціями категорії *Visualization* цільової платформи. Якщо ця функціональність підтримана в обраній цільовій платформі, то для конфігурації сигнальної системи використовується об'єкт *Alarm configuration* на вкладці *Resources*. На рис. 4.7 зображено вікно налаштування конфігуратора тривог.

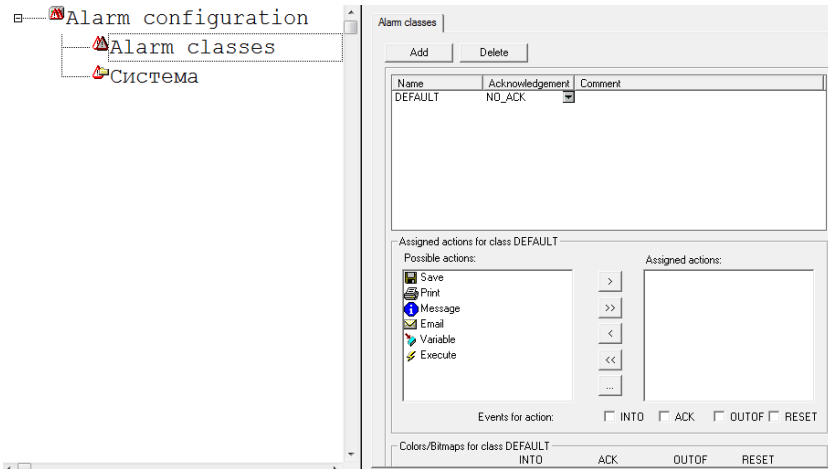


Рис. 4.7 – Вікно конфігуратора тривог

Дерево конфігурації складається з класів тривог та груп тривог. Клас тривог визначає параметри, які властиві даному виду тривоги. Група тривог відповідає конкретній конфігурації однієї або декількох тривог, які використовуються в проекті. Таким чином, класи корисні для структурування тривог. Різні групи тривог формує користувач, вставляючи відповідні розділи під головним заголовком **System** в дереві конфігурації.

Для візуалізації тривог в *CDS2* передбачений спеціальний елемент *Alarm table* (Таблиця тривог). Використовуючи таку таблицю, ко-

ристувач може спостерігати та підтверджувати тривоги. Якщо необхідна історія, тобто запис тривожних подій, повинен бути заздалегідь визначений *log*-файл і для кожної групи визначені параметри запису.

Для подальшого налаштування об'єкта *Alarm configuration* розглянемо деякі терміни та визначення.

Так, **Alarm** – це подія, зумовлена певними умовами (значенням виразу). Кожна тривога має **Priority**, який визначає її ступень важливості. Також тривоги розрізняються за станом – **Alarm state**. Цей стан може мати наступні значення: **NORM** – тривоги немає, **INTO** – тривога тільки що відбулася, **ACK** – тривога відбулася та підтверджена користувачем, **OUTOF** – умови тривоги зникли, але вона не підтверджена. Тривоги можуть мати проміжний стан **Sub-State**, тобто умови тривоги можуть включати межі (**Lo** – нижній, **Hi** верхній) та надзвичайні межі (**LoLo**, **HiHi**).

Головна мета тривог полягає в тому, щоб повідомити користувачеві про критичні ситуації. При цьому часто необхідно упевнитися, що користувач помітив цю інформацію (можливі дії по тривозі задані в конфігурації класу). Користувач повинен підтвердити тривогу, щоб видалити її зі списку. Це так зване підтвердження тривог – **Acknowledgement of alarms**.

Тривожна подія – **Alarm Event**, це той момент, при якому фіксуємо тривогу або зміна її стану. У конфігурації тривог *CDS2* для трьох типів тривожних подій і відповідних станів застосовуються однакові назви (**INTO**, **ACK**, **OUTOF**).

Класи тривог використовуються для опису деяких загальних критеріїв тривог, типу, способу підтвердження користувачем, дій, які повинні автоматично виконуватися по тривожним подіям, квітів і картинок, що використовуються для візуалізації Таблиці тривог. Класи тривог визначені глобально в конфігурації тривог та доступні при конфігуруванні груп тривог.

Для створення нового класу тривог натисніть ЛКМ на кнопку **Add**. Слідом за цим у верхньому вікні буде доданий рядок, спочатку з єдиним елементом **NOACK** (без підтвердження) в колонці **Acknowledgement**. Визначте назву для нового класу у відповідному полі колонки **Name**. Якщо необхідно, змініть тип підтвердження. Можливі такі варіанти підтвердження:

- **NO_ACK**: підтвердження не потрібно;
- **ACK_INT0**: виникнення умов тривоги (статус **INT0**) повинен бути підтверджений користувачем;

– **ACK_OUTOF**: зникнення умов тривоги (статус **OUTOF**) повинен бути підтверджений користувачем;

– **ACK_ALL**: виникнення та зникнення умов тривоги повинен бути підтверджений користувачем.

Для кожного класу тривог може бути вибрана дія, яка виконується за тривожною подією. Тривожні події можуть бути такими:

– **INTO**, виникла тривога (**Status = INTO**).

– **ACK**, підтвердження виконано користувачем (**Status = ACK**).

– **OUTOF**, умови тривоги зникли (**Status = OUTOF**).

Конфігурування груп тривог здійснюється у вкладенні **Alarm group**. За допомогою кнопки **Add** в групу додається рядок з параметрами тривоги. Для тривоги визначаються наступні параметри:

– **Expression**, це вираз, який складений із змінних проекту, за яким оцінюються умови тривоги;

– **Type**, це тип тривоги з перерахованих нижче:

- **DIG = 0**, дискретна тривога, активна поки значення виразу має статус **FALSE**;

- **DIG = 1**, дискретна тривога, активна поки значення виразу має статус **TRUE**;

- **LOLO**, аналогова тривога, активна поки значення виразу нижче межі **Alarm type LOLO**;

- **LO**, аналогова тривога, яка подібна **LOLO**;

- **HI**, аналогова тривога, активна поки значення виразу вище межі **Alarm type HI**;

- **HHI** аналогова тривога, яка подібна **HI**;

- **DEV-**, відхилення в менший бік від заданої величини. Тривога активна якщо значення виразу нижче заданої величини, визначеної для **Alarm type DEV-**. Відхилення задається у відсотках;

- **DEV+**, відхилення в більший бік від заданої величини. Тривога активна, якщо значення виразу вище заданої величини, визначеної для **Alarm type DEV+**.

- **ROC**, швидкість зміни параметра. Тривога стає активною, як тільки значення виразу починає змінюватися з певною швидкістю. Межа формування тривоги визначає величина зміни **Rate of changes** в одиницю часу: в секунду, хвилину або годину;

– **Class**, перелік класів;

– **Priority**, пріоритет тривоги в діапазоні 0-152, де 0 - це найвищий пріоритет;

– **Message**, повідомлення тривоги. Цей текст буде з'являтися у вікні повідомлення. Однак натискання кнопки **OK** в цьому вікні не формує підтвердження користувача. Підтвердження здійснюється через таблицю тривоги.

– **Deactivation**, логічна змінна проекту деактивує будь-яке створення тривоги.

4.4. Перевірка роботи навчального проекту

1) Перевірте роботу програми користувача в проекті. Для цього вмикайте перемикачі та вводите значення локальних змінних і спостерігайте за всіма змінами в програмі користувача на екрані монітора. Зафіксуйте візуалізацію зробивши декілька скриншотів. Подайте проект викладачу для перевірки.

2) Складіть звіт в редакторі *Microsoft Office* відповідно до правил оформлення звітів: відомості про виконавця, назву та мету роботи, схему з'єднань, лістинг програми ПЛК, параметри налаштувань *COM*-порту для завантаження проектів та скриншоти екранів з візуалізаціями в режимі **Online**. Для документування проекту використовуйте вбудовану в *CDS2* утиліту **Document** в меню **Project**.

4.5. Завдання для самостійної роботи

1) Розробіть проект 2-позиційного регулятора на основі ФБ **ON_OFF_HIST_REG** з бібліотеки **PID_Rerulators.lib**.

2) Розробіть проект ПІД-регулятора з використанням ФБ **PID_3POS_IM_ANR** для керування 3-позиційним виконавчим пристроєм з аналоговим сигналом керування.

3) Розробіть проект ПІД-регулятора з використанням ФБ **PID_FUNCTION** та ФБ **VALVE_REG**.

4.6. Контрольні питання

1) Дайте коротку характеристику принципів ПІД-регулювання.

2) Яким чином можна реалізувати ПІД-регулювання за допомогою дискретних входів ПЛК?

3) Як враховуються характеристики виконавчих пристроїв під час управління клапаном без датчика положення та з датчиком положення?

ЗМІСТ

Вступ	3
<i>Практичне завдання 1.</i> Загальні принципи програмування контролерів <i>ОВЕН</i> в середовищі <i>CODESYS V2.3</i>	5
<i>Практичне завдання 2.</i> Структурування проектів та основні принципи використання таймерів та лічильників	41
<i>Практичне завдання 3.</i> Принципи реалізації 2-позиційного та ШІМ регулювання.	51
<i>Практичне завдання 4.</i> Принципи реалізації ПІД-регулювання. .	66
Список літератури	79