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

КОМП'ЮТЕРНІ СИСТЕМИ КЕРУВАННЯ ТЕРМОХІМІЧНИМИ ТА ЕЛЕКТРОХІМІЧНИМИ ПРОЦЕСАМИ ВИДОБУТКУ ВОДНЮ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня магістра
за освітньою програмою «Інжиніринг інтелектуальних електротехнічних та мехатронних комплексів»
спеціальності 141 Електроенергетика, електротехніка та електромеханіка

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2025

Укладачі: *Торопов А.В.*, канд. техн. наук., доцент
Босак А.В., канд. техн. наук., доцент
Торопова Л.В.
Поліщук В.О.

Рецензент: *Чернявський А.В.*, канд. техн. наук., доцент
кафедра електропостачання

Відповідальний редактор *Кулаковський, Л.Я.*, канд. техн. наук, доцент

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 5 від 06.03.2025 р.)
за поданням Вченої ради навчально-наукового інституту енергозбереження та енергоменеджменту
(протокол № 6 від 30.01.2025 р.)

Комп'ютерні системи керування термохімічними та електрохімічними процесами видобутку водню: комп’ютерний практикум[Електронний ресурс]: навч. посіб. для здобувачів ступеня магістра за освітньою програмою «Інжиніринг інтелектуальних електротехнічних та мехатронних комплексів» / А.В. Торопов, А.В. Босак, Л.В. Торопова, В.О. Поліщук; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,5 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2025. – 59с.

У представленому посібнику викладено основні положення щодо виконання комп’ютерних практикумів, тематика яких обіймає розділи курсу по вивченню процедури програмування логічних контролерів, розробки систем візуалізації процесів автоматизації і контролю. Навчальне видання призначене для здобувачів ступеня магістра за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка», освітньою програмою «Інжиніринг інтелектуальних електротехнічних та мехатронних комплексів»

Реєстр. № НП 24/25-283. Обсяг 1,3 авт. арк.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

З М І С Т

Вступ

Дисципліна «Комп'ютерні системи керування термохімічними та електрохімічними процесами видобутку водню» є однією із завершальних у професійній підготовці інженерів, що навчаються за сертифікатною програмою «Інжиніринг та автоматизація водневих енергетичних систем і технологій». Вона є логічним продовженням таких спеціальних дисциплін як «Теорія автоматичного керування», «Автоматизація технологічних процесів», «Автоматизація управління промисловими об’єктами» та ін.

Основна мета комп'ютерних практикумів – дати студентам ґрунтовні навички щодо проектування та налагодження сучасних систем автоматизації електрохімічного процесу видобутку водню на базі програмованих логічних контролерів. В результаті виконання роботи студенти повинні набути знань про загальні особливості побудови схеми автоматизації та вибору її компонентів, налаштування параметрів позиційних та ПІД – регуляторів, а також розробки високоефективного людино-машинного інтерфейсу для роботи із системою із підвищеною небезпекою. Навчальне видання до виконання комп'ютерних практикумів містить основні теоретичні відомості, вказівки для вирішення практичних задач та контрольні запитання. Для самостійної підготовки студенти користуються конспектом лекцій та рекомендованою літературою.

Комп'ютерний практикум №1. Керування засувкою подачі електроліту.

Тривалість заняття – 2 години

Одним із найбільш розповсюджених виконавчих механізмів в технологічних процесах є засувка. Існує дуже широка класифікація засувок як за принципом роботи, так і за способом управління. Розрізняють:

- засувки із ручним приводом, керування якими в автоматичному режимі неможливе, проте іноді в зовнішню систему управління подається сигнал про поточне положення засувки;
- засувки із електричним приводом, що керується дискретними (релейними) сигналами на відкриття та закриття;
- засувки з електричним приводом, що керується аналоговими сигналами (0..10В або 4..20мА), в якому ступінь відкриття засувки є пропорційним вхідному сигналу.

З точки зору реалізації керування найбільш складним є керування електричним приводом за допомогою релейних сигналів. На відміну від клапанів, що мають лише два положення, в такої засувки ступінь відкриття визначається часом подачі керуючих сигналів. Суттєву похибку в точність регулювання засувкою вносить інерційність електроприводу та наявність люфту виконавчого механізму, що із часом збільшується. Відповідно виникає необхідність корекції розрахованого положення засувки за допомогою кінцевих вимикачів. Крім того, слід враховувати інерційність засувки, тобто засувка з електричним приводом змінює положення не миттєво, а протягом хвилини – двох.

Найпростішим способом реалізації засувки є використання лічильника, що рахує імпульси на вході і, в залежності, від напряму лічби зменшує або збільшує вихідне значення. В Codesys для цього може бути використаний реверсивний блок лічильника CTUD, що має окремі входи лічби на збільшення та зменшення вихідного значення. Далі за допомогою генератора імпульсів BLINK бібліотеки Util.lib, в залежності від команд відкриття та закриття засувки, подаються імпульси або на вхід збільшення лічби, або на вхід зменшення. Шляхом встановлення тривалості і частоти імпульсів може бути врахований час відкриття і закриття засувки.

Для зручності роботи із функціональними блоками використовуємо мову програмування Continious Flow Chart (CFC) з можливістю вільного переміщення блоків. Блок-схема керування засувкою мовою CFC представлена на рисунку 1.1.

Частота подачі імпульсів, й відповідно швидкість відкриття/закриття засувки встановлюється параметром T1, який в нашому випадку виставлений

T#500ms (1Гц). Повний час відкриття засувки виставляємо рівним 60 секундам, який подаємо на вхід PV лічильника.

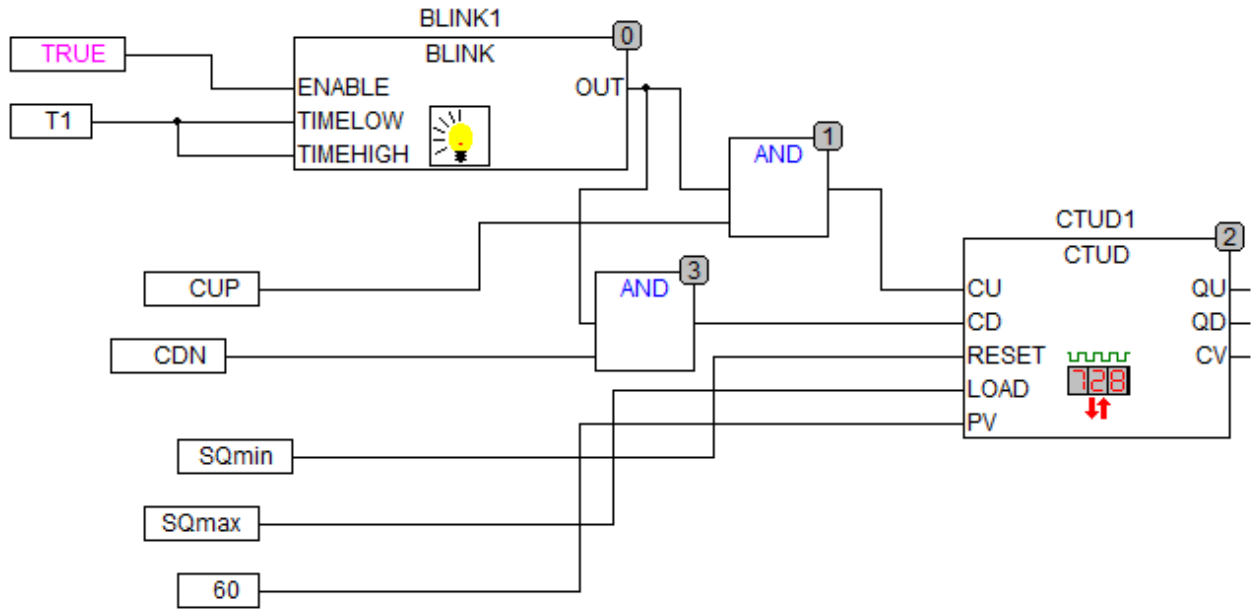


Рисунок 1.1 – Блок-схема керування засувкою.

Сигнали SQmin і SQmax дозволяють врахувати похибку обчислення поточного положення засувки. Так, при спрацюванні кінцевого вимикача, що формує сигнал SQmin, контролер визначає, що він знаходиться в положенні «повністю закрито». Аналогічно, при отриманні сигналу SQmax контролер дізнається, що засувка є повністю відкритою.

Для формування імпульсів відкриття/закриття засувки сформуємо програму мовою програмування ST. Для цього використаємо структуру IF, ELSIF, END_IF, тобто команди умовного переходу.

```
IF FIset>FIact THEN VALVE.CUP:=TRUE; VALVE.CDN:=FALSE;
ELSIF FIset<FIact THEN VALVE.CUP:=FALSE; VALVE.CDN:=TRUE;
ELSE VALVE.CUP:=FALSE; VALVE.CDN:=FALSE;
END_IF;
```

Поточне значення положення засувки згідно алгоритму, представленому на рисунку 1.1 визначається вихідним сигналом поточного значення лічильника CTUD:

```
FIact:=VALVE.CTUD1.CV;
```

Далі запускаємо в основному тілі програми виклик підпрограми емуляції засувки:

```
VALVE();
ACTUAL_Position_VALVE:=VALVE.CTUD1.CV;
K1:=VALVE.CUP; (*Команда вмикання відповідного реле на відкриття*)
K2:=VALVE.CDN; (*Команда вимикання відповідного реле на закриття*)
```

Контрольні запитання:

1. За допомогою яких блоків бібліотеки Util.lib може бути реалізований генератор імпульсів?
2. Які мови програмування контролерів використовуються при написанні програми керування засувкою в даному прикладі?
3. Навіщо в програмі здійснюється обробка сигналів кінцевих вимикачів?
4. Які ви знаєте типи лічильників в Codesys?
5. Чому в даній програмі для визначення часу роботи використовується лічильник, а не таймер?

Комп'ютерний практикум №2. Налаштування цифрової фільтрації вимірювань

Тривалість заняття – 2 години.

Для додаткового захисту від електромагнітних перешкод у цифрових пристроях окрім аналогової фільтрації передбачено фільтрацію на рівні програмного забезпечення для низьких частот.

Цифрова фільтрація зазвичай проводиться у два етапи.

На першому етапі фільтрації з поточних вимірів вхідних параметрів відфільтровуються значення, що мають явно виражені провали або викиди. Для цього контролер обчислює різницю між результатами вимірювань вхідної величини, виконаних у двох останніх циклах опитування, і порівнює її із заданим значенням, так званою смугою фільтра. Якщо обчислена різниця перевищує задану межу, що виставлена в параметрах налаштування фільтра, то проводиться повторний вимір, отриманий результат відкидається, а значення лінії фільтра подвоюється. В разі підтвердження нового значення фільтр перебудовується (тобто смуга фільтра зменшується до вихідної) на новий стабільний стан вимірюваної величини. Такий алгоритм дозволяє захистити прилад від впливу поодиноких імпульсних і комутаційних перешкод, що виникають на виробництві під час роботи силового устаткування.

На другому етапі фільтрації здійснюється згладжування (демпфування) сигналу з метою усунення шумових складових. Найпростіше це зробити за рахунок обчислення середнього арифметичного за останніми декількома вимірюваннями. При цьому, чим більша кількість останніх вимірювань, що використовуються при обчисленні середнього значення, тим глибшим буде фільтрування вхідного сигналу. Ефект фільтрації можна побачити з графіків, представлених на рисунку 2.1.

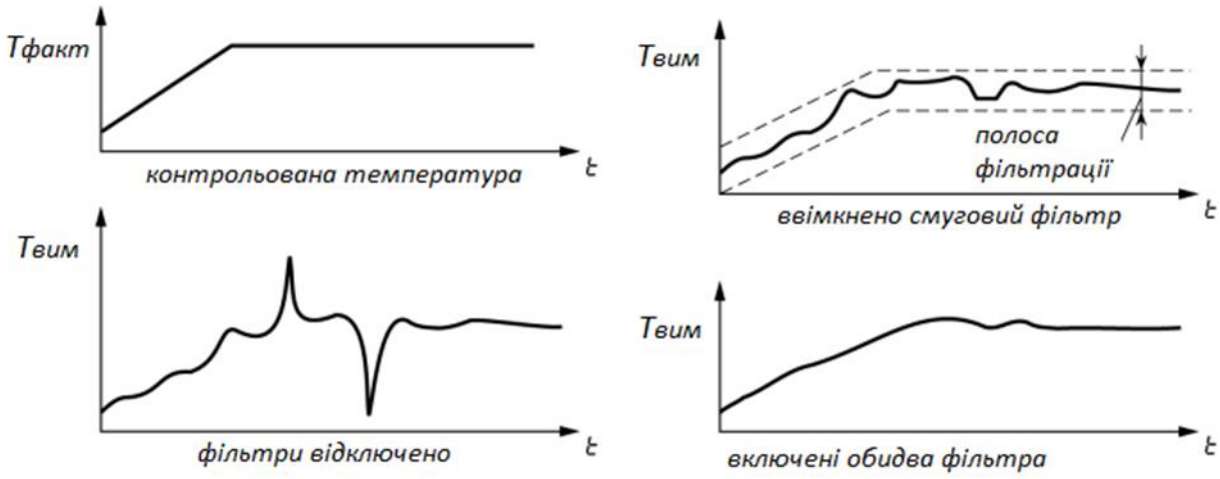


Рисунок 2.1 - Ефект фільтрації аналогового значення.

Програмна реалізація цифрової фільтрації у програмному середовищі Codesys.

Почнемо розробку фільтру із другої частини, а саме розробки підпрограми демпфування аналогового сигналу. Слід зазначити, що для забезпечення фіксованої сталої часу фільтру, що не залежить від об’єму програми, необхідно здійснювати опитування аналогового входу періодично із фіксованим часом опитування. При незначних об’ємах програмного коду контролера допустиме використання готових функціональних блоків «генератор імпульсів» та «детектор фронту». Використовуючи мову програмування CFC (Continuous Flow Chart) та блоки бібліотеки UTILS.lib можна реалізувати генератор імпульсів, які, в свою чергу, можна прив’язати до програми опитування аналогового сигналу. Програма матиме вигляд, представлений на рисунку 2.2.

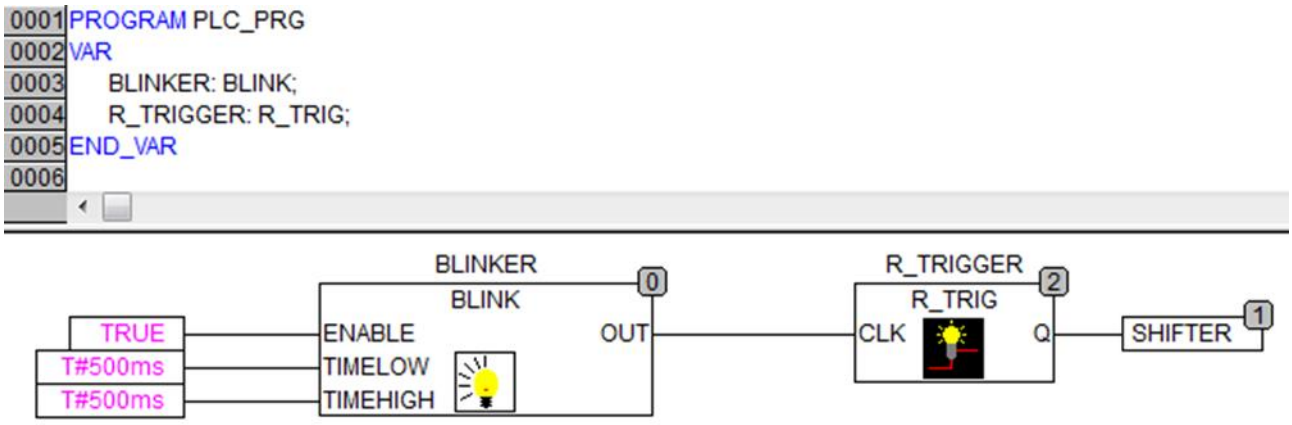


Рисунок 2.2 - Програма генерування імпульсів SHIFTER.

Слід відзначити, що змінну SHIFTER необхідно призначити глобальною VAR_GLOBAL, щоб до неї могли звертатися системні події та інші програми.

Далі необхідно налаштувати чергу виконання задач, а саме програмних блоків в ресурсному полі «Конфігурація задач», що представлено на рисунку 2.3.

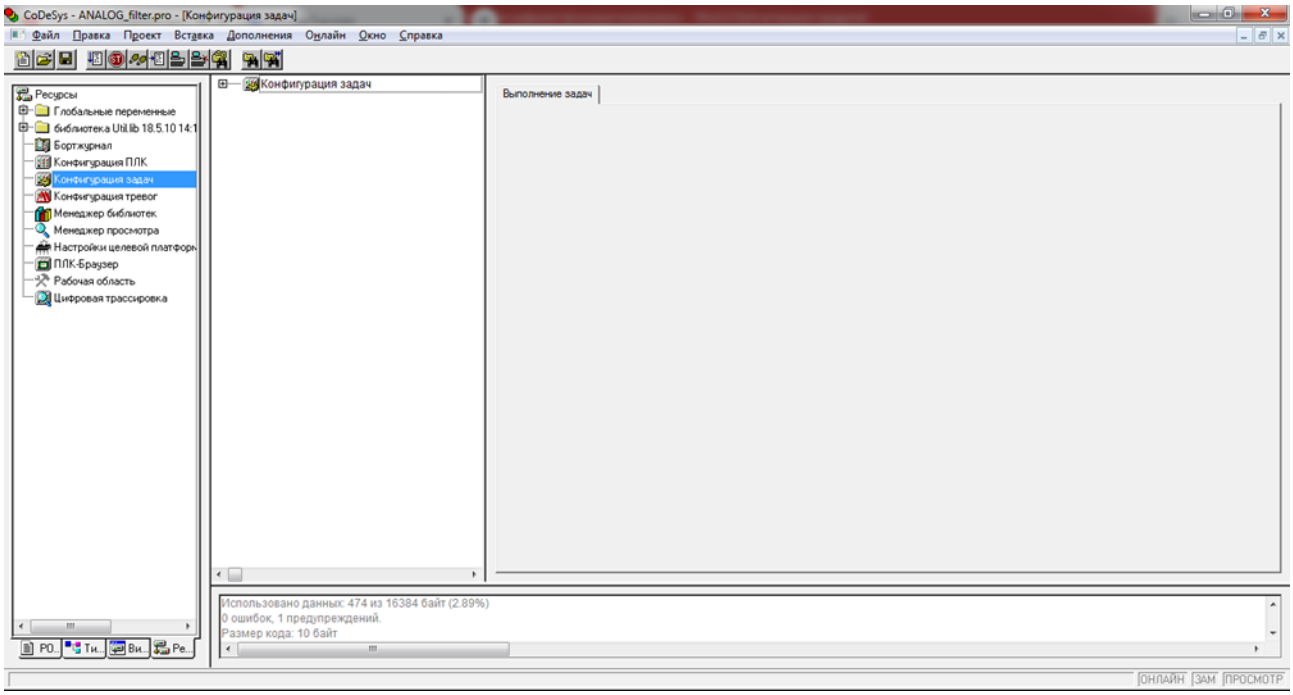


Рисунок 2.3 - Конфігуратор задач в полі ресурсів проекту.

Для цього попередньо в дереві проекту створимо нову програму, що буде написана мовою програмування ST, як це можна побачити на рисунку 2.4.

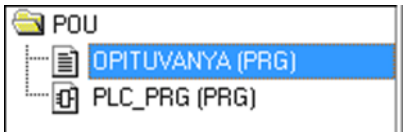


Рисунок 2.4 - Program organization units поточної програми.

Далі в налаштуванні конфігуратора задач створюємо дві задачі для запуску та даємо їм для зручності назви, що співпадають із назвою програм, що виконуються, як це представлено на рисунку 2.5.

Рисунок 2.5 - Конфігуратор задач із двома активованими задачами.

Властивості задачі OPITUVANNA представлені на рисунку 2.6.

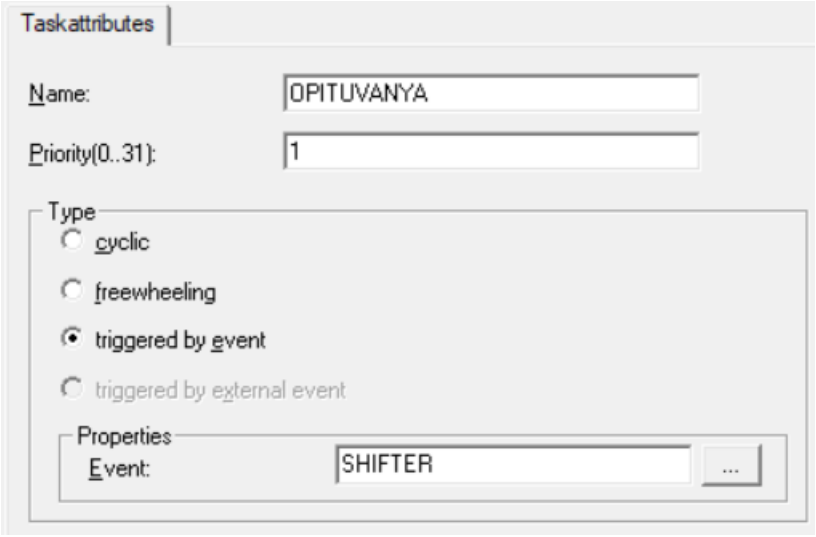


Рисунок 2.6 - Властивості задачі опитування програми OPITUVANYA.

Програма має найвищий пріоритет з тих, що використовуються для вирішення цієї задачі, а запуск виконання програми здійснюється за подією, а саме SHIFTER=TRUE.

Основна програма PLC_PRG має властивості, представлені на рисунку 2.7.

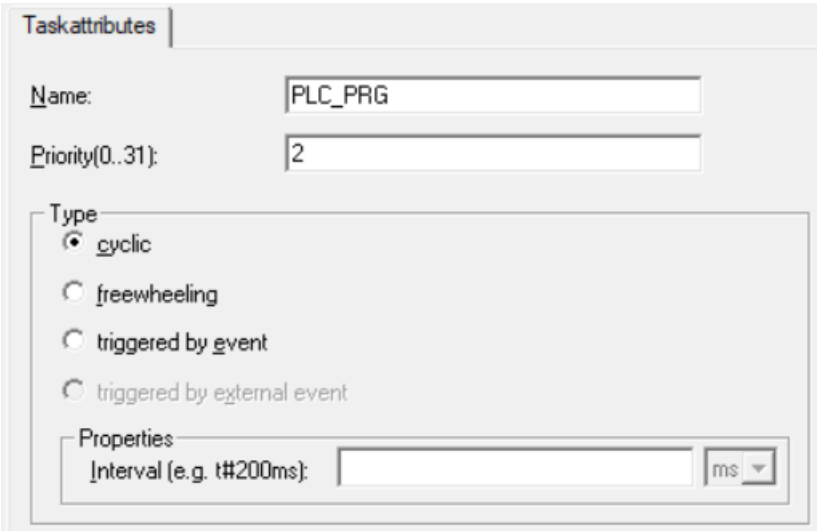


Рисунок 2.7 - Властивості задачі опитування програми PLC_PRG.

Програма PLC_PRG виконується циклічно, отже з кожним новим циклом контролера, проте із більш низьким пріоритетом в роботі, ніж команда OPITUVANYA.

Далі налаштовуємо програму опитування та фільтрації аналогового сигналу. При цьому програму реалізуємо текстовою мовою програмування ST.

Визначимо основні змінні, що використовуються в програмі OPITUVANYA.

```
PROGRAM OPITUVANYA
VAR
  A: ARRAY [0..8] OF INT;
  AI1: REAL;
  A1fact: REAL;
  i: WORD;
  SUMA: INT;
  DEEP_FILTER: INT := 7;
  INIT: INT:=7;
  BAND: INT := 30;
  Kfilter: INT:=1;
END_VAR
```

Масив «A» є масивом останніх значень змінної, що вимірюється;

AI1 – поточне значення аналогового сигналу, що отримує контролер з аналогового входу;

A1fact – відфільтроване значення аналогового сигналу;

DEEP_FILTER – глибина демпфування, що визначає кількість останніх вимірювань, що використовуються для обчислення середнього значення;

BAND – смуга пропускання, тобто величина максимального відхилення змінної, що буде враховуватися;

Kfilter - коефіцієнт множення смуги пропускання, що подвоюється при кожному вимірюванні за межами смуги пропускання.

Основна частина програми OPITUVANYA має вигляд:

```
(*Початкове обчислення значення на основі першого вимірювання*)
WHILE INIT<>0 DO
A[INIT-1]:=REAL_TO_INT(AI1);
INIT:=INIT-1;
END_WHILE;
(*Розташування змінних в області пам'яті за принципом перший ввійшов - останній вийшов*)
IF ABS(A1fact-AI1)<Kfilter*BAND THEN
A[7]:=A[6];
A[6]:=A[5];
A[5]:=A[4];
A[4]:=A[3];
A[3]:=A[2];
A[2]:=A[1];
A[1]:=A[0];
A[0]:=REAL_TO_INT(AI1);
(*Обчислення середнього значення для заданої кількості останніх вимірювань*)
FOR i:=0 TO DEEP_FILTER DO
SUMA:=SUMA+A[i];
END_FOR;
(*Множення на коефіцієнт є процедурою аналогічне діленню*)
A1fact:=SUMA*1/INT_TO_REAL(DEEP_FILTER+1);
SUMA:=0; Kfilter:=1;
ELSE Kfilter:=2*Kfilter;
(*Команда подвоєння при отриманні сигналу, що не водить в смугу пропускання фільтру*)
END_IF;
```

Контрольні запитання:

1. Які існують програмні способи фільтрації аналогових сигналів в програмованих логічних контролерах?
2. Що таке переривання і для чого вони використовуються?
3. Для чого в програмі використовується функціональний блок R_TRIG?
4. На що впливає глибина фільтрації сигналу?
5. Яким чином в програмі здійснюється відсікання хибного сигналу?

Комп'ютерний практикум №3. Реалізація функції мотогодин напрацювання водневої установки із використанням блоку Statistics.

Тривалість заняття – 2 години.

У кожну машину, вузол та агрегат конструкторами закладено певний ресурс роботи. Він залежить від різних факторів як внутрішніх, так і зовнішніх. Це якість металу, параметрів електроживлення, умови роботи, режими експлуатації та багато іншого. Для обладнання за одиницю ресурсу прийнято визначати мотогодинами. Зазвичай підрахунок мотогодин реалізується на механічному лічильнику або на мікропроцесорному пристрої із збереженням кількості мотогодин в енергонезалежну пам’ять.

Теоретичні відомості про модуль статистики (Statistic).

Модуль статистики (Statistic) призначений для видачі в програму користувача інформаційних даних про функціонування ОВЕН ПЛК:

- наявність/відсутність живлення – відображається у бітовому каналі Power status (живлення – «1»);
- інформація про помилки, що виникають під час роботи ПЛК, – відображається в бітовому каналі Error («0» – ПЛК працює штатно, «1» – відбулася помилка при функціонуванні ПЛК);
- три рядки показників роботи ПЛК: значення циклу роботи ПЛК в сотнях мкс (Cycle time in 100 mks), дозволяє користувачеві оцінити обсяг обчислювальних ресурсів, який потрібний для роботи написаної ним програми. Якщо цикл виявляється більше, заданого в параметрах роботи ПЛК параметра MinCycleLength, це вказує на те, що програма користувача занадто вимоглива до ресурсів, і цей параметр бажано збільшити, щоб цикли не перекривалися;
- час, що залишився до вимкнення ПЛК, сек (Time to backup power down, s), відображає ресурс часу роботи ПЛК від акумуляторних батарей (з відключеним зовнішнім живленням). Характеристика є приблизною, точне врахування впливу всіх факторів (температура ПЛК та зовнішнього середовища, точність вимірювання та ін.) є досить складним. У разі роботи ПЛК від мережі акумуляторна батарея заряджається, і цей параметр опосередковано вказує на процес заряджання (600 с відповідає повністю зарядженій батареї);
- температура всередині ПЛК (Temp inside PLC), що відображає температуру, заміряну датчиком усередині корпусу ПЛК (у різних моделей ПЛК температура може вимірюватися на різних платах, визначається інтенсивністю нагріву конкретних плат). Характеристика опосередковано свідчить про стан ПЛК.

Особливістю біту Power Status є те, що він ідентифікує наявність нормального живлення, тобто при зникненні живлення спочатку біт дорівнює нулеві, а вже потім відбувається вимикання контролера та обнулення змінних. Аналогічно при вмиканні біт стає рівним одиниці вже після ініціалізації всіх параметрів.

Оскільки постійне присвоювання змінній мотогодин нового значення призведе до затирання енергонезалежної пам'яті, доцільно зберігати значення мотогодин в енергонезалежну пам'ять тільки при зникненні живлення ПЛК.

Здійснимо прив'язку до контролера ОВЕН серії ПЛК100, в якому реалізована така функція. Для цього скачуємо файл прив'язки до входів/виходів ПЛК за посиланням:

https://aqteck.com.ua/uploads/110/install_target_v2.12_for_plc1xx.zip

Натискаємо Install file для папки відповідного контролера, наприклад ПЛК100.К.-L.

У вікні, що відкрилося, натискаємо Open і далі вибираємо plc.tnf. Натискаємо Install, при цьому Target-файл відобразиться у установлених.

Далі відкриваємо Codesys 2.3 і вибираємо Target – файл PLC100.K.L, що розташований в кінці списку.

Спочатку додаємо модуль Statistics, в якому реалізовано функцію контролю напруги. Для цього заходимо в конфігурацію контролера та вибираємо відповідний модуль при натисканні правої клавіші, як це представлено на рисунку 3.1.

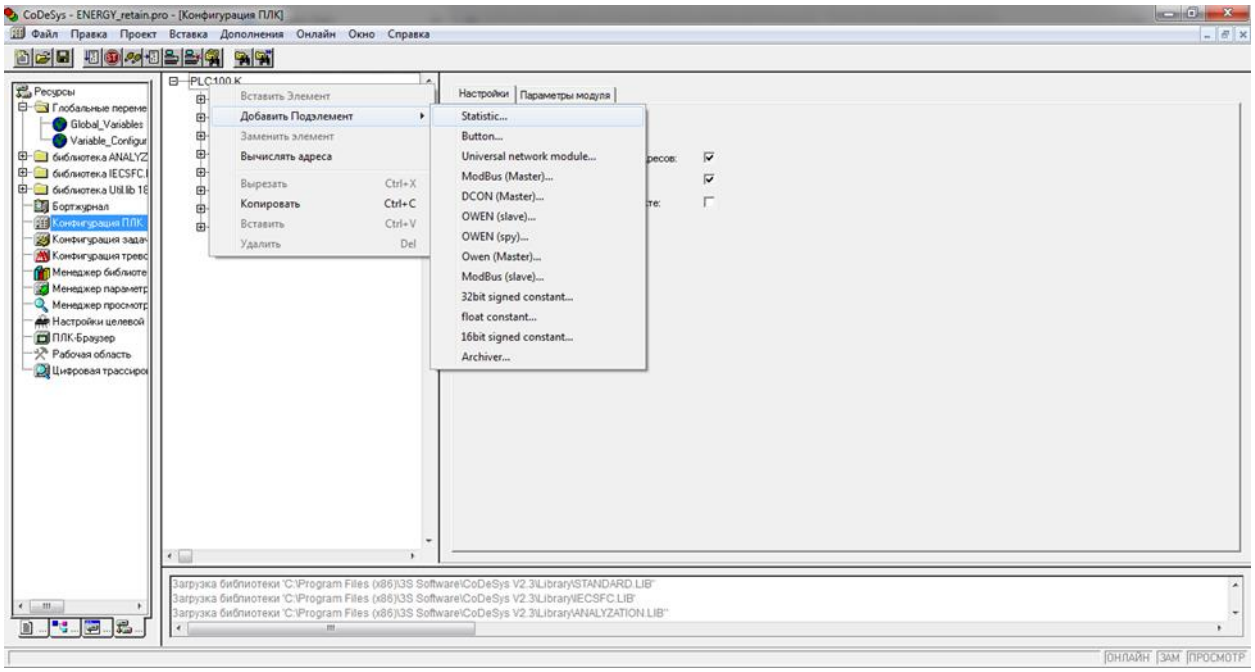


Рисунок 3.1 – Активація модуля Statistics

При натисканні лівої клавіші миші відкриваються змінні цього модуля, як це можна бачити на рисунку 3.2.

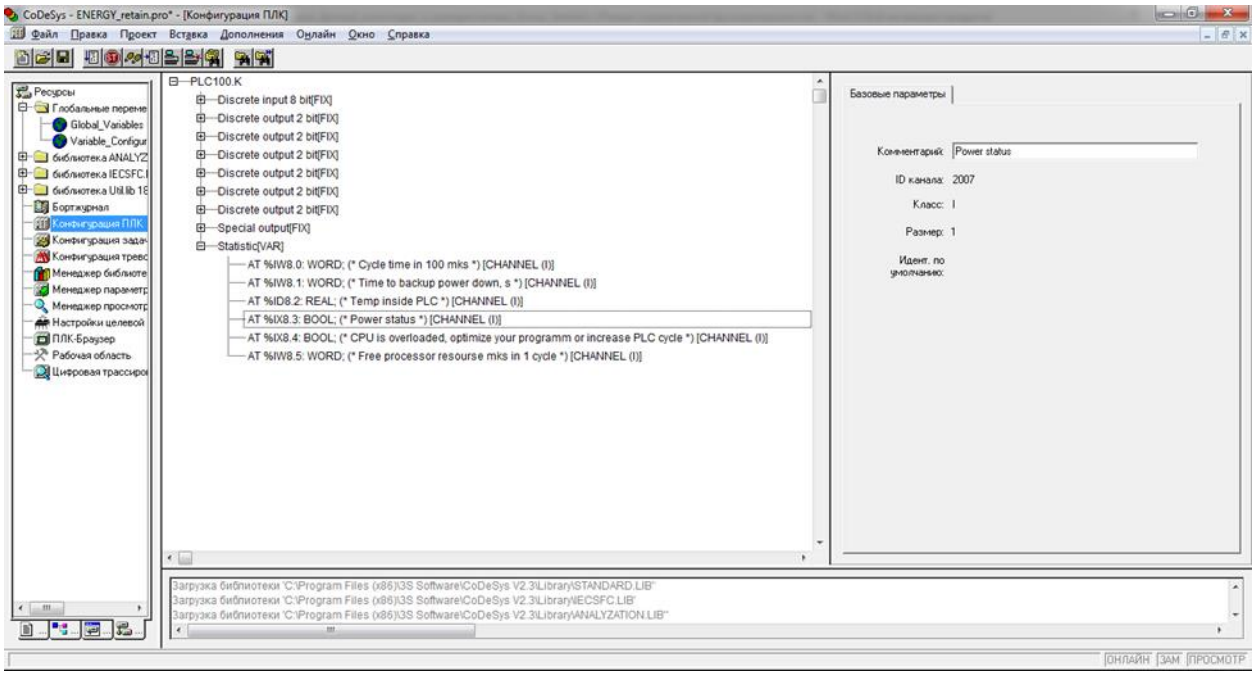


Рисунок 3.2 – Змінні модуля Statistics

Біт контролю наявності живлення знаходиться за адресом %IX8.3.

Для емуляції спрацювання біту наявності живлення в полі глобальних змінних об’являємо нову змінну:

```
 VAR_GLOBAL
  M1 AT %IX8.3: BOOL;
END_VAR
```

Спочатку напишемо функцію генератора імпульсів з періодом 1 секунда, що активується від біту наявності живлення:

```
BLINKER(ENABLE:=M1  ,TIMELOW:=T#500ms,  TIMEHIGH:=T#500ms  , OUT=> );
```

Далі напишемо частину програми, що відповідає за зчитування поточного значення мотогодин та запис цього значення при виникненні та зникненні живлення:

```
R_TRIGGER(CLK:=M1 , Q=> );
IF R_TRIGGER.Q=TRUE THEN MOTOCLOCK:=RETAIN_MC;
END_IF;
F_TRIGGER(CLK:=M1 , Q=> );
IF F_TRIGGER.Q=TRUE THEN RETAIN_MC:=MOTOCLOCK;
END_IF;
```

Змінні MOTOCLOCK та RETAIN_MC описуємо в форматі DWORD, це дозволить отримувати значення мотогодин до 4294967295 секунд, що відповідає 49710 дням (при використанні WORD обмежуємось 17 днями напрацювання).

При спрацюванні переднього фронту M1, тобто виникненні живлення значення з RETAIN_MC копіюється у MOTOCLOCK, а при зникненні M1 з MOTOCLOCK копіюється у RETAIN_MC.

Далі, оскільки змінна типу DWORD немає можливості використовувати готовий лічильник типу CTU, отже пишемо лічильник вручну.

```
R_TRIGGER2(CLK:=BLINKER.OUT , Q=> );
IF R_TRIGGER2.Q=TRUE THEN MOTOCLOCK:=MOTOCLOCK+1;
END_IF;
```

По кожному імпульсу генератора імпульсів з періодом 1 секунда, величина MOTOCLOCK збільшується на одиницю.

Для емуляції вимикання контролера введемо змінну POWER_LOW:

```
IF POWER_LOW THEN MOTOCLOCK:=0;
END_IF;
```

Відзначимо, що змінні позначаємо як:

```
PROGRAM PLC_PRG
VAR
  R_TRIGGER: R_TRIG;
  MOTOCLOCK: DWORD;
  BLINKER: BLINK;
  F_TRIGGER: F_TRIG;
  COUNTUP: CTU;
  R_TRIGGER2: R_TRIG;
  POWER_LOW: BOOL;
END_VAR
```

```
VAR RETAIN (*позначення енергонезалежних змінних*)
  RETAIN_MC: DWORD;
END_VAR
```

В середовищі програмування Codesys дана програма має вигляд, представлений на рисунку 3.3.

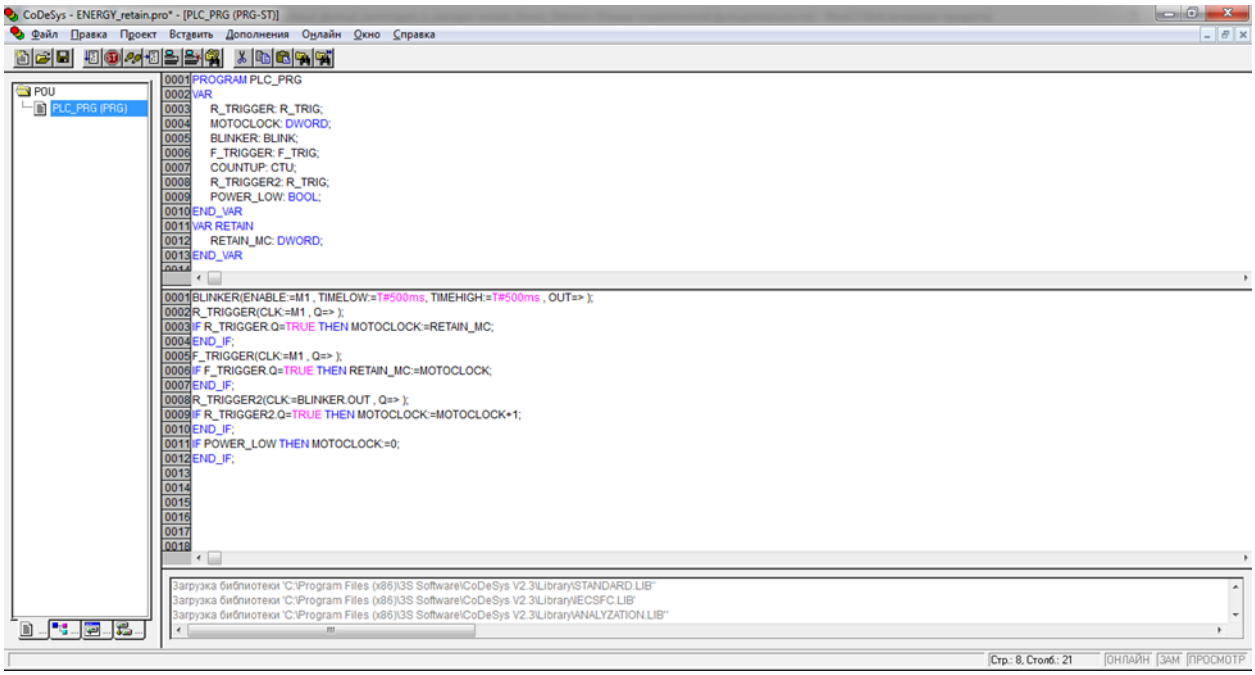


Рисунок 3.3 – Вікно програми.

Здійснюємо перевірку працездатності програми. Встановлюємо галочку емуляції та запускаємо проект.

Натискаємо один раз на квадратне поле біту живлення, отримуємо зміну стану, представлену на рисунку 3.4.

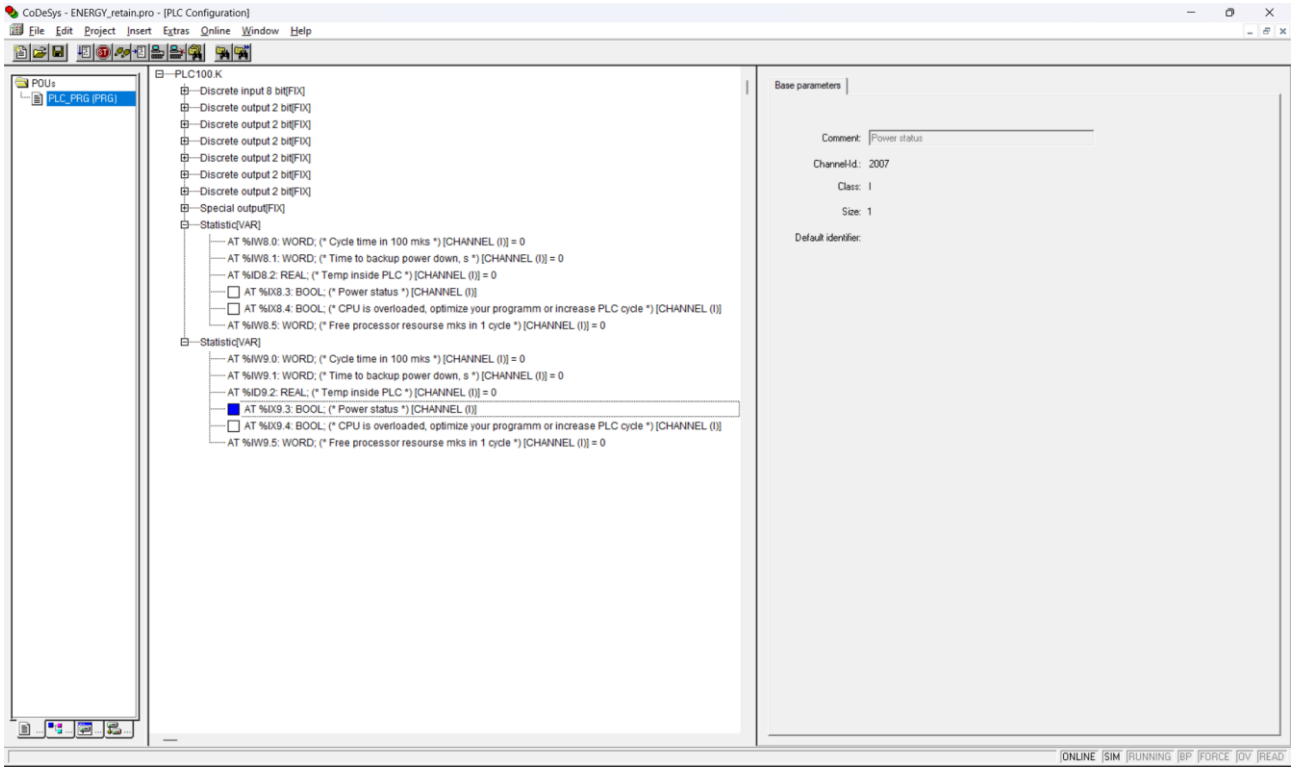


Рисунок 3.4. Керування змінними модуля Statistics.

Повертаємось в основну програму, перевіряємо, чи йде збільшення MOTOCLOCK. Потім повертаємось до блоку статистики і прибираємо біт напруги. При цьому лічильник повинен зупинитися. Присвоюємо POWER_LOW значення TRUE, а потім FALSE. Це скине значення MOTOCLOCK. При появі біту напруги значення лічильника повинно копіюватися з RETAIN_MC і збільшуватися далі.

Контрольні запитання:

1. Що таке лічильник мотогодин і для чого він використовується?
2. Який функціонал передбачений в модулі Statistics?
3. З яким періодом здійснюється опитування часу напрацювання обладнання в даному прикладі?
4. Вкажіть адресу біту наявності живлення.
5. В чому проблема постійного запису поточного запису значення мотогодин в енергонезалежну пам'ять контролера?

Комп'ютерний практикум №4. Емуляція технологічних процесів з використанням користувацьких функціональних блоків.

Тривалість заняття – 2 години.

При використанні бібліотеки Util.lib, а саме розділу математичних функцій з'являється можливість емулювати поведінку електрохімічного чи теплохімічного процесу.

Так, в розділі математичних функцій наявні наступні функціональні блоки:

- DERIVATIVE (*функціональний блок для обчислення похідної від значення вхідної змінної, причому для обчислення завжди використовуються чотири значення для максимального точного визначення похідної *)
- INTEGRAL (* функціональний блок для обчислення інтеграла від змінної IN за часом *)
- LIN_TRAFO (*функціональний блок лінійного перетворення*)
- STATISTICS_INT (*функціональний блок для статистики: обчислення мінімуму, максимуму та середнього значення вхідного значення змінної типу INT*)
- STATISTICS_REAL (*функціональний блок для статистики: обчислення мінімуму, максимуму та середнього значення вхідного значення змінної типу REAL*)
- VARIANCE (*функціональний блок для обчислення математичної дисперсії вхідної змінної*).

Використовуючи процедуру заміни коливальних та аперіодичних ланок інтегральними ланками, що охоплені зворотними зв'язками фактично можна отримати наближену модель будь-якого технологічного процесу із видобутку водню. Фактично слід частково здійснити перехід від математичної моделі у виді передаточних функцій до моделі у вигляді простору станів.

Розглянемо даний підхід на прикладі аперіодичної ланки першого порядку:

$$\frac{X_{\text{вих}}}{X_{\text{вх}}} = \frac{K}{Tp+1}$$

Переходячи до нормальної форми Коші отримає вираз для вихідної змінної в явному вигляді:

$$X_{\text{вих}} = \frac{1}{p}\left(-\frac{1}{T}X_{\text{вих}} + \frac{K}{T}X_{\text{вх}}\right).$$

Відповідно, при подачі на вхід інтегратора масштабованих вхідного та вихідного сигналів, на виході інтегруючої ланки отримуємо зміну сигналу за аперіодичним законом.

Для зручності відображення взаємозв'язків при запису вищезазначеного рівняння, використовуємо мову програмування CFC. Налаштування нового функціонального блоку представлені на рисунку 4.1.

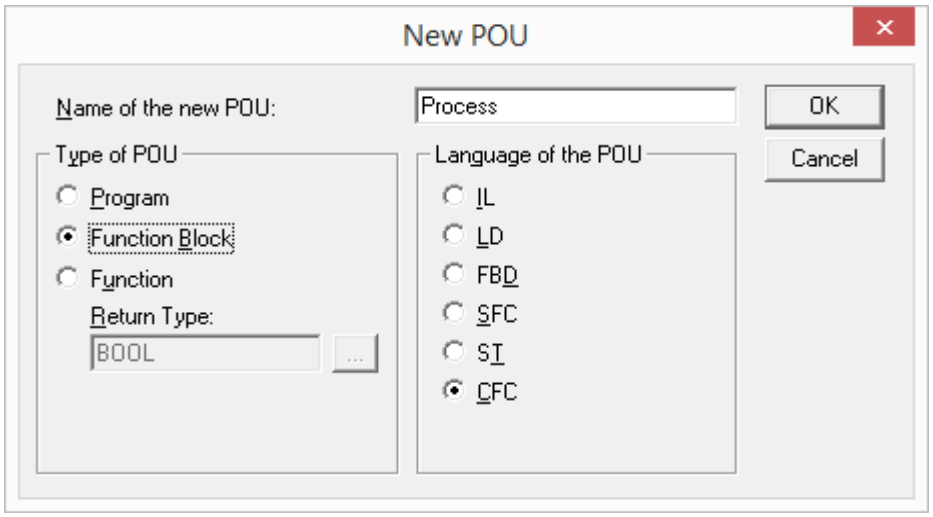


Рисунок 4.1 - Налаштування функціонального блоку аперіодичного процесу.

В редакторі Library Manager вибираємо бібліотеку Util.lib і активуємо її. При цьому автоматично довантажується й бібліотека Standard.lib, як це можна побачити на рисунку 4.2.

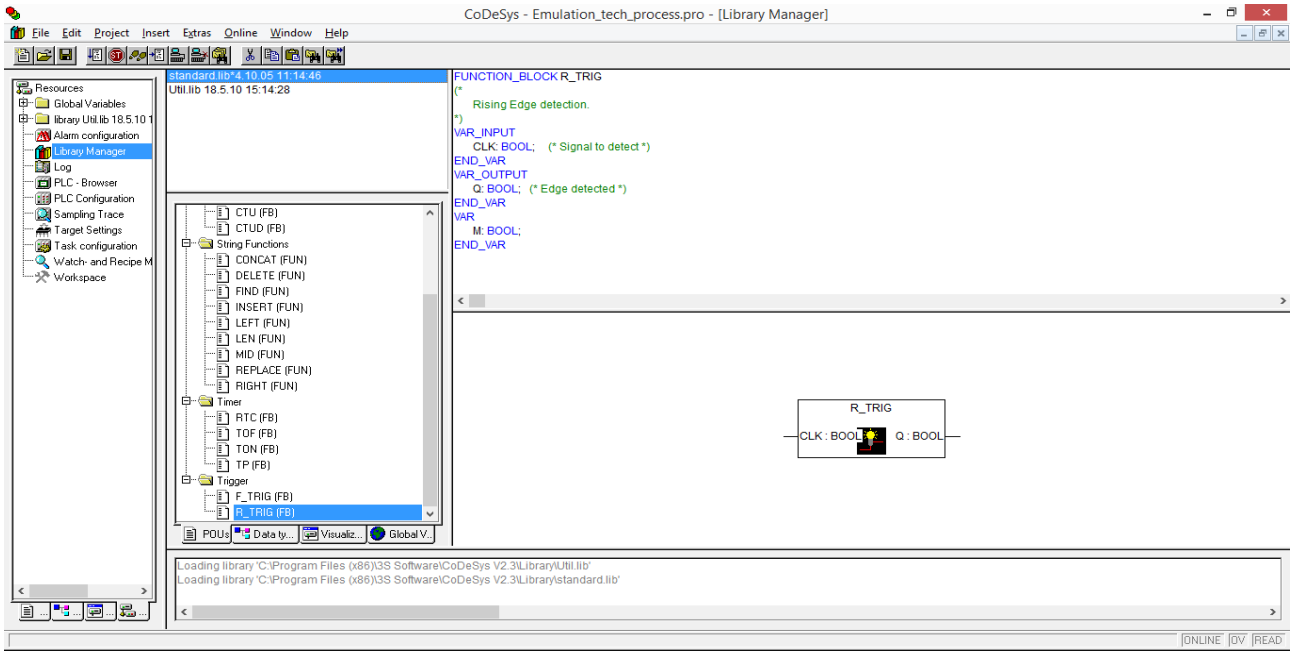


Рисунок 4.2 - Підключення бібліотек для реалізації технологічного процесу.

Далі у вкладці Process (FB) набираємо структурну схему, що відповідає рівнянню Коші для аперіодичної ланки. Спочатку вибираємо функціональний блок інтегрування, як це представлено на рисунку 4.3.

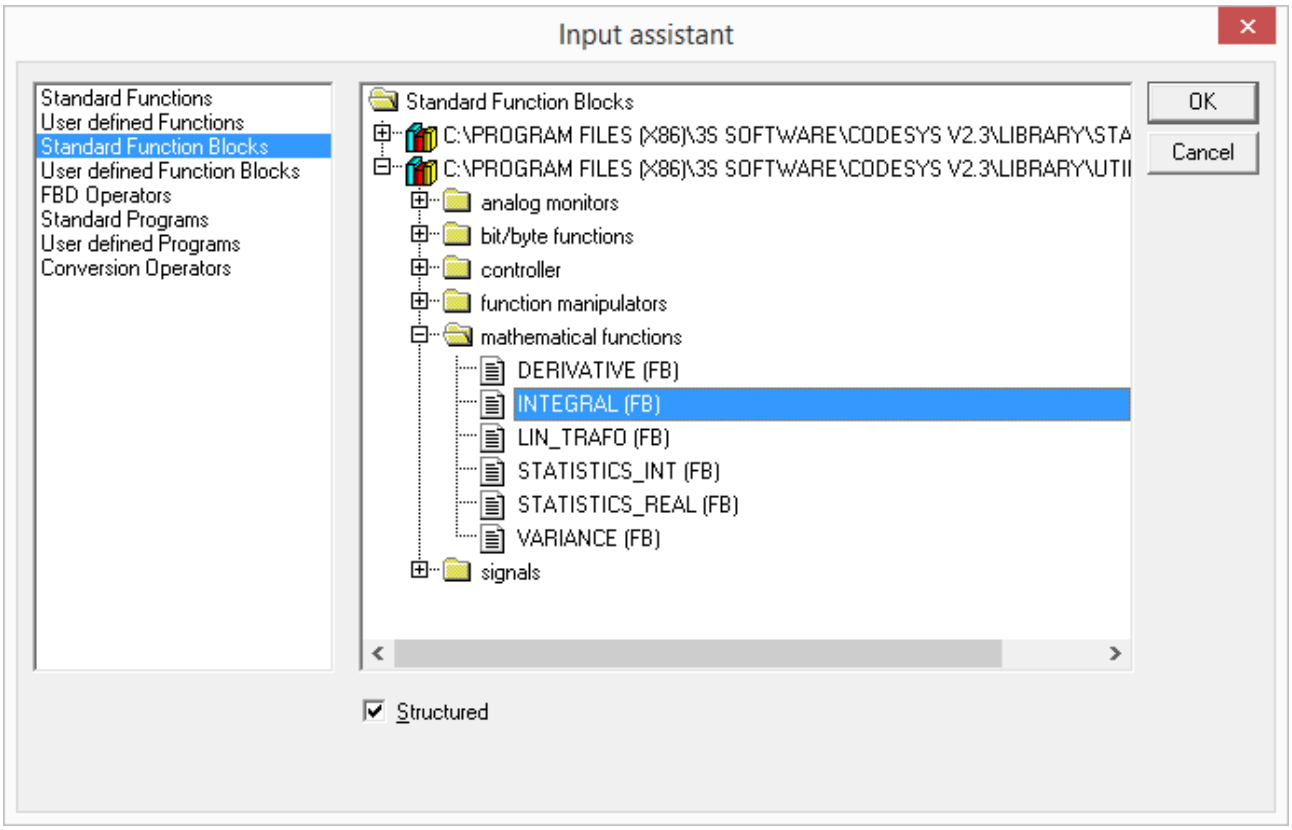


Рисунок 4.3 - Вибір блоку інтегрування.

Створюємо структуру мовою CFC, що відповідає рівнянню аперіодичної ланки в формі Коші, що представлена на рисунку 4.4.

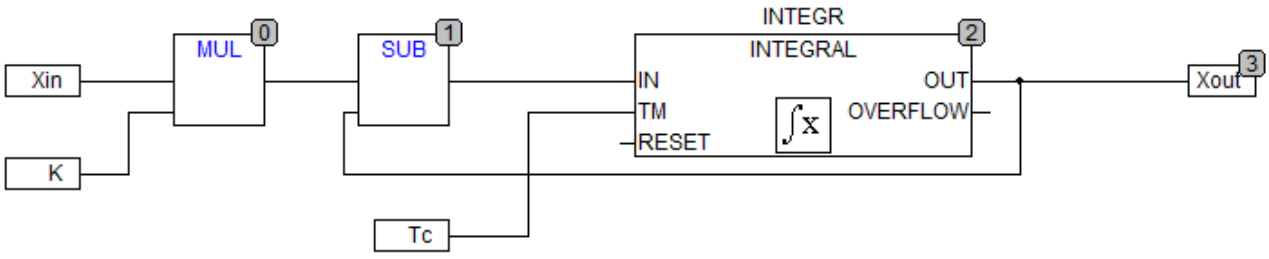


Рисунок 4.4 - Структурна схема аперіодичної ланки.

При цьому всі вхідні, вихідні та проміжні змінні позначаємо наступним чином:

```
 FUNCTION_BLOCK Process
VAR_INPUT
  Xin: REAL;
END_VAR
VAR_OUTPUT
  Xout: REAL;
END_VAR
VAR
  INTEGR: INTEGRAL;
  K: REAL := 1; (*Коефіцієнт підсилення)
  Tc: DWORD:=10; (*Час виклику процесу інтегрування в мс, має зворотний вплив на сталу інтегрування T*)
END_VAR
```

Далі в основній програмі PLC_PRG вибираємо користувацький блок Process(FB), як це представлено на рисунку 4.5.

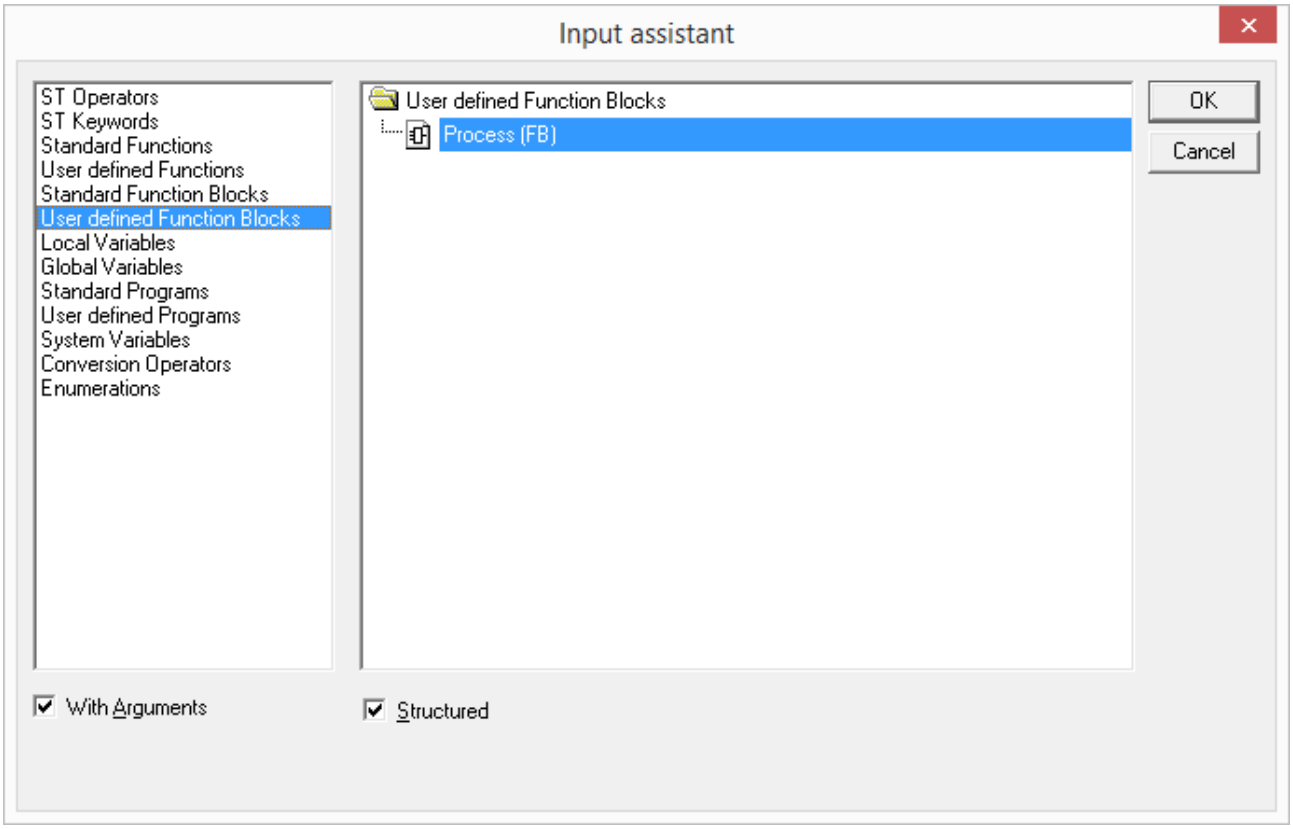


Рисунок 4.5 - Вибір користувацького функціонального блоку.

Додаємо ім’я цьому блоку і прописуємо для нього вхідні та вихідні змінні:

Process_A(Xin:=IN1 , Xout=>OUT1);

Для вибраних змінних і блоків формуємо позначення:

```
PROGRAM PLC_PRG
VAR
  IN1: REAL;
  OUT1: REAL;
  Process_A: Process; (*Вибираємо в користувацьких блоках*)
END_VAR
```

Для відображення перехідного процесу створюємо блок візуалізації і в ньому підключаємо повзунок для завдання сигналу і тренд для його відображення.

Налаштування повзунка робимо наступними, як представлено на рисунку 4.6.

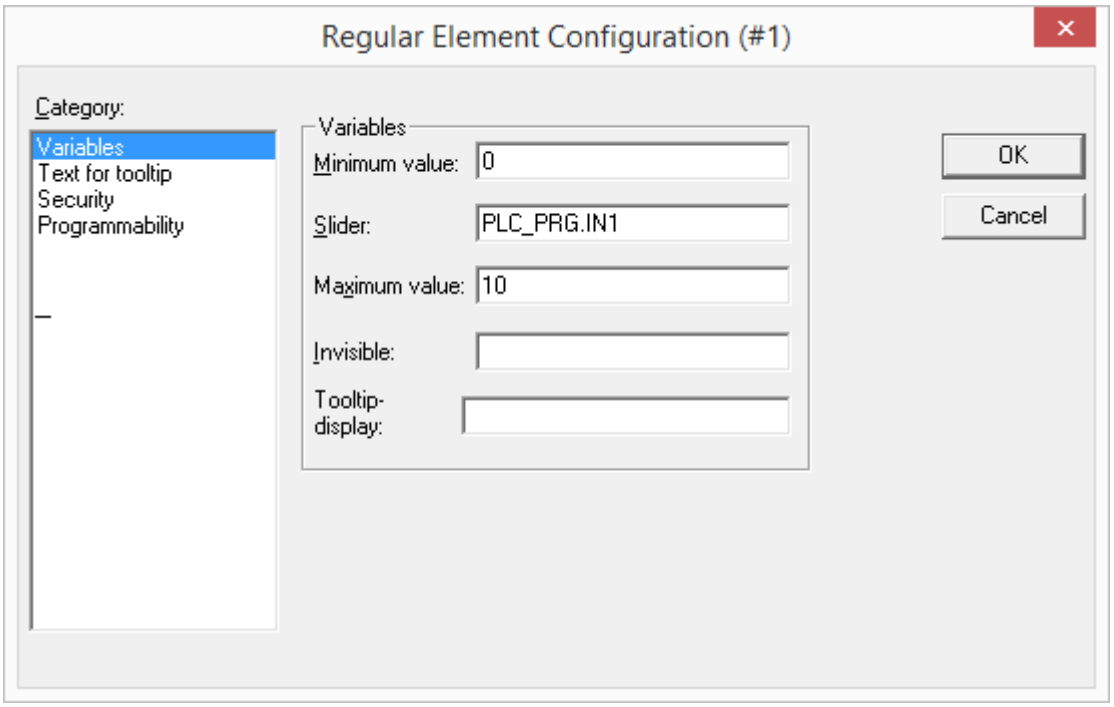


Рисунок 4.6 - Налаштування повзунка.

Налаштування горизонтальної вісі тренду представлені на рисунку 4.7.

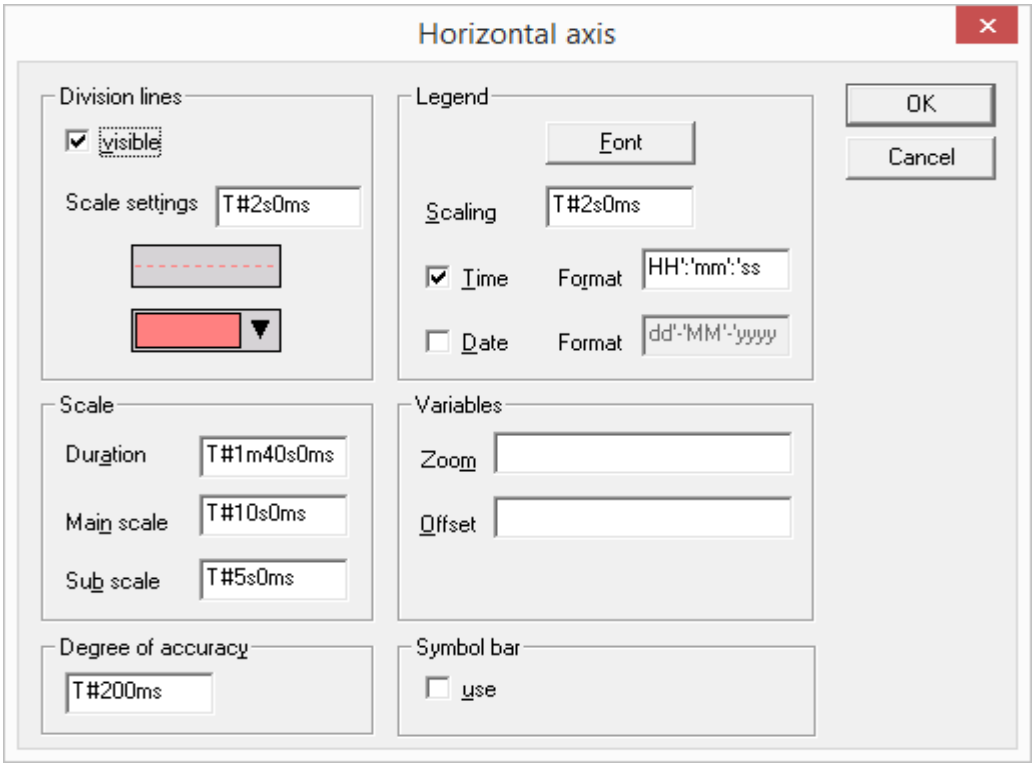


Рисунок 4.7 - Налаштування горизонтальної вісі.

Крім цього, в налаштуваннях Choose variable вибираємо в якості змінної графіку OUT1.

Після переводу повзунка в положення 10 та подачі команди RUN, графік перехідного процесу має вигляд, представлений на рисунку 4.8.

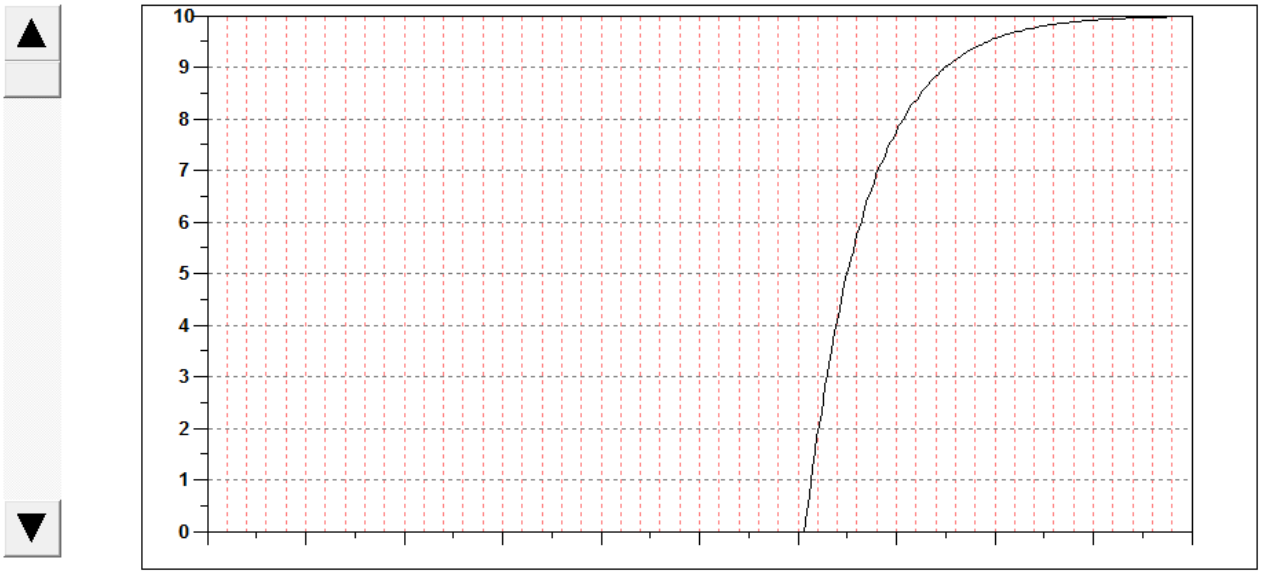


Рисунок 4.8 - Характер зміни аперіодичного перехідного процесу при реалізації рівняння в формі Коші.

Контрольні запитання:

1. Які функціональні блоки є в розділі математичних функцій бібліотеки Util.lib?
2. Який характер перехідного процесу має місце в системі для даного прикладу?
3. В якій формі представлений опис процесу в даному прикладі?
4. Яким чином в налаштуваннях здійснюється визначення змінних функціонального блоку, що будуть доступними в основній програмі?
5. Назвіть в даному прикладі ціну однієї поділки між вертикальними червоними лініями.

Комп'ютерний практикум №5. Побудова математичної моделі технологічних процесів із чистим запізнюванням.

Тривалість заняття – 2 години.

Більшість технологічних процесів регулювання температури містять чисте запізнювання керуючого сигналу. Відповідно, для налаштування регуляторів та формування еталонних моделей необхідно вміти реалізовувати запізнювання сигналу в середовищі Codesys.

Для початку сформуємо сигнал, що буде змінюватися у часі за визначеним алгоритмом. Для цього найкращим є блок генерації сигналів GEN бібліотеки Util.lib, представлений на рисунку 5.1.

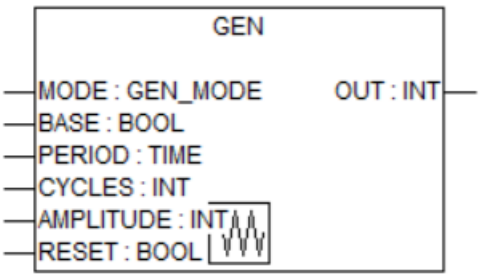


Рисунок 5.1 - Зовнішній вигляд генератора імпульсів.

Параметри налаштування блоку генерації сигналів:
-MODE:GEN_MODE;
(TRIANGLE – трикутна форма сигналу від -AMPL до +AMPL; TRIANGLE_POS - трикутна форма сигналу від 0 до AMPL; SAWTOOTH_RISE - пилкоподібна форма сигналу, що наростає від -AMPL до +AMPL;
SAWTOOTH_FALL - пилкоподібна форма сигналу, що спадає від AMPL до -AMPL;
RECTANGLE – меандр із пермиканням від -AMPL до +AMPL;
SINUS – синусоїдальний сигнал;
COSINUS – косинусоїдальний сигнал);
- BASE:BOOL;
(* FALSE: період визначається за змінною CYCLES;
TRUE: період визначається за змінною PERIOD);
- PERIOD:TIME:=t#1s (активно за умови BASE=TRUE);
- CYCLES:INT:=1000 (активно за умови BASE= FALSE);
- AMPLITUDE:INT (максимальне значення вихідного сигналу);

- RESET:BOOL (скидання вихідного сигналу в нуль).

Відповідно сформуємо код, що відповідає роботі задатчика синусоїди:

```
GEN_SINUS(MODE:=SINUS,BASE:=FALSE
,PERIOD:=T#1ms,CYCLES:=1000, AMPLITUDE:=100 ,RESET:= ,OUT=> );
```

Далі формуємо масив комірок пам’яті для змінних типу REAL і прив’язуємо вихід генератора до першого значення:

```
A1[0]:=GEN_SINUS.OUT;
```

Для рівномірності розподілу змінних протягом часу в комірках пам’яті використовуємо генератор меандрового сигналу із стробуючими імпульсами за переднім фронтом:

```
BLINK1(ENABLE:=TRUE ,TIMELOW:=T#99ms ,TIMEHIGH:=T#1ms
,OUT=>);
R_TRIG1(CLK:=BLINK1.OUT , Q=> );
```

Далі за кожним імпульсом даємо команду перезапису всіх поточних значень в наступну комірку. При цьому реалізуємо внутрішній цикл FOR:

```
IF R_TRIG1.Q=TRUE THEN (*По передньому фронту стробуючого
імпульсу записати цикл перезапису всіх змінних*)
FOR i:=0 TO 998 DO
A1[999-i]:=A1[998-i];
END_FOR;
END_IF;
```

Число 999 визначає максимальну поточну кількість комірок для реалізації функції затримки. Із вибраними періодом генерації імпульсів (100мс) 2е відповідає 99,9с. При необхідності це число може бути збільшене.

На заключному етапі вибираємо поточне значення комірки, з якої буде зчитуватися вихідна координата згідно алгоритму:

```
Xout:=A1[DELAY];
```

При реалізації даної програми в середовищі Codesys 2.3, скріншот програми має вигляд, представлений на рисунку 5.2.

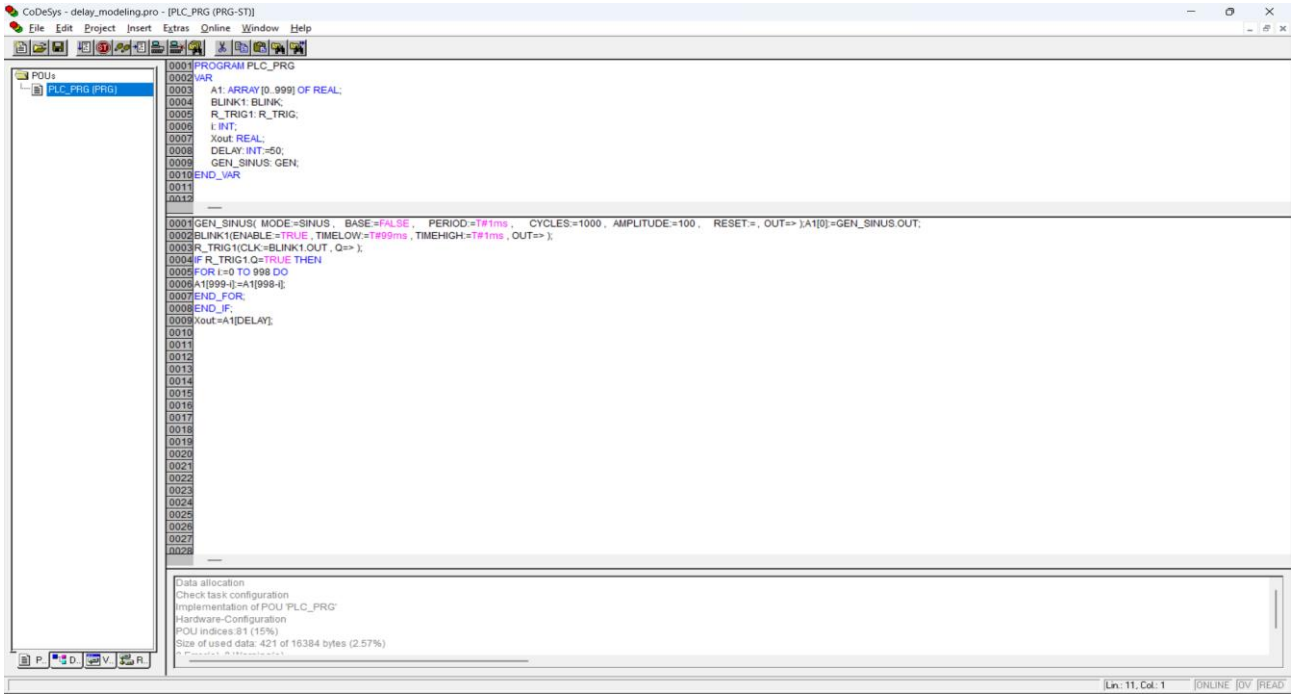


Рисунок 5.2 - Зовнішній вигляд скріншота програми.

Далі необхідно вивести графіки виходу генератора синусоїдального сигналу та вихідної змінної. Для цього використовуємо вбудовану візуалізацію. Для відображення вищевказаних графіків використовуємо блок Trend. Далі у вікні налаштувань графіків вибираємо змінні, що мають відображатися. Також вказуємо налаштування для кожної змінної, що представлені на рисунку 5.3.

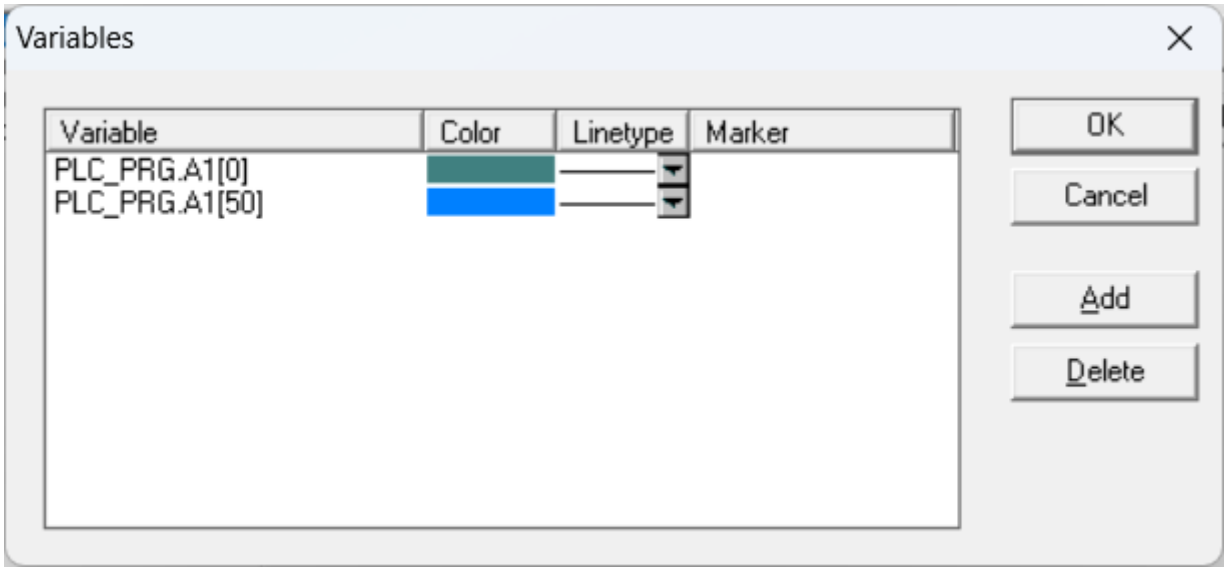


Рисунок 5.3 - Налаштування параметрів, графіки яких виводяться для візуалізації.

В налаштуваннях графіків вибираємо розмірність осей по X та Y координатах в полі Axis, як це можна побачити на рисунку 5.4.

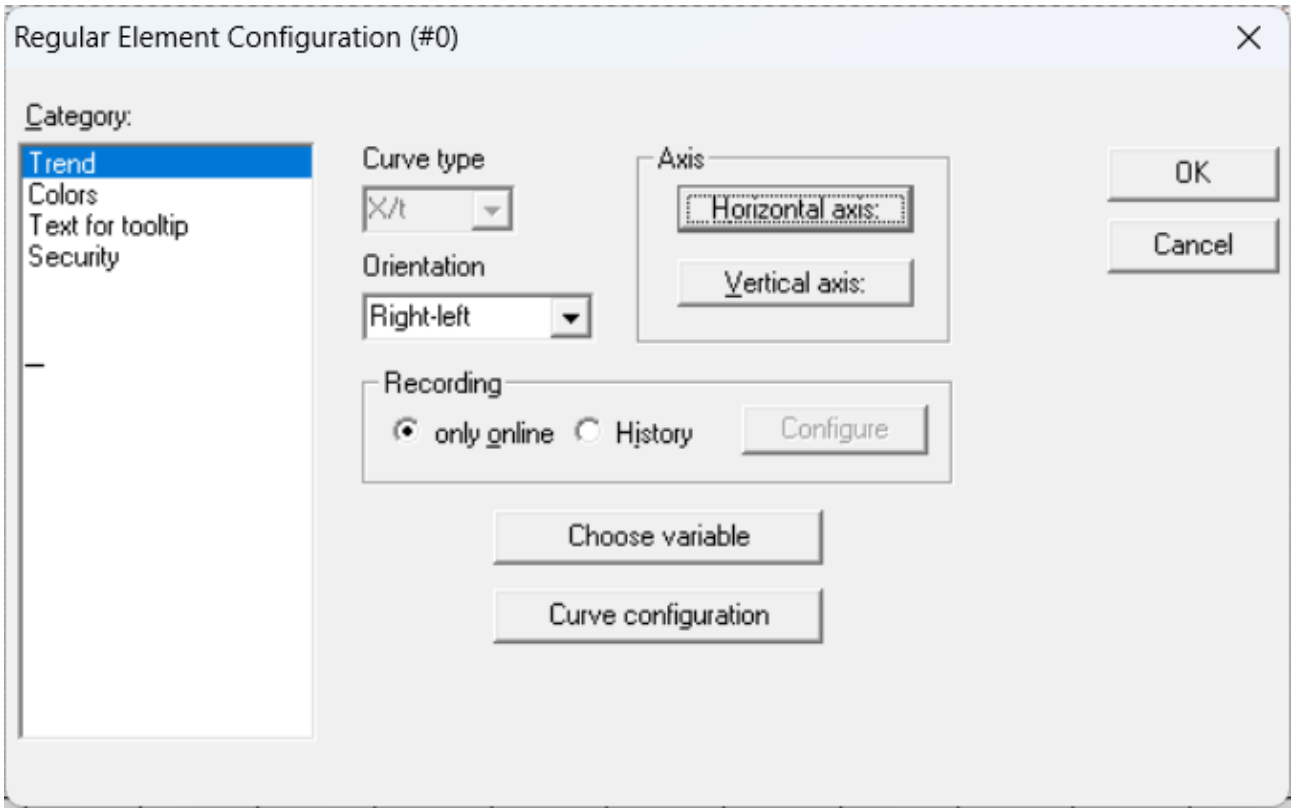


Рисунок 5.4 - Налаштування графіків.

Після запуску симуляції шляхом вибору Online та Login на екран візуалізації виводяться графіки, представлені на рисунку 5.5.

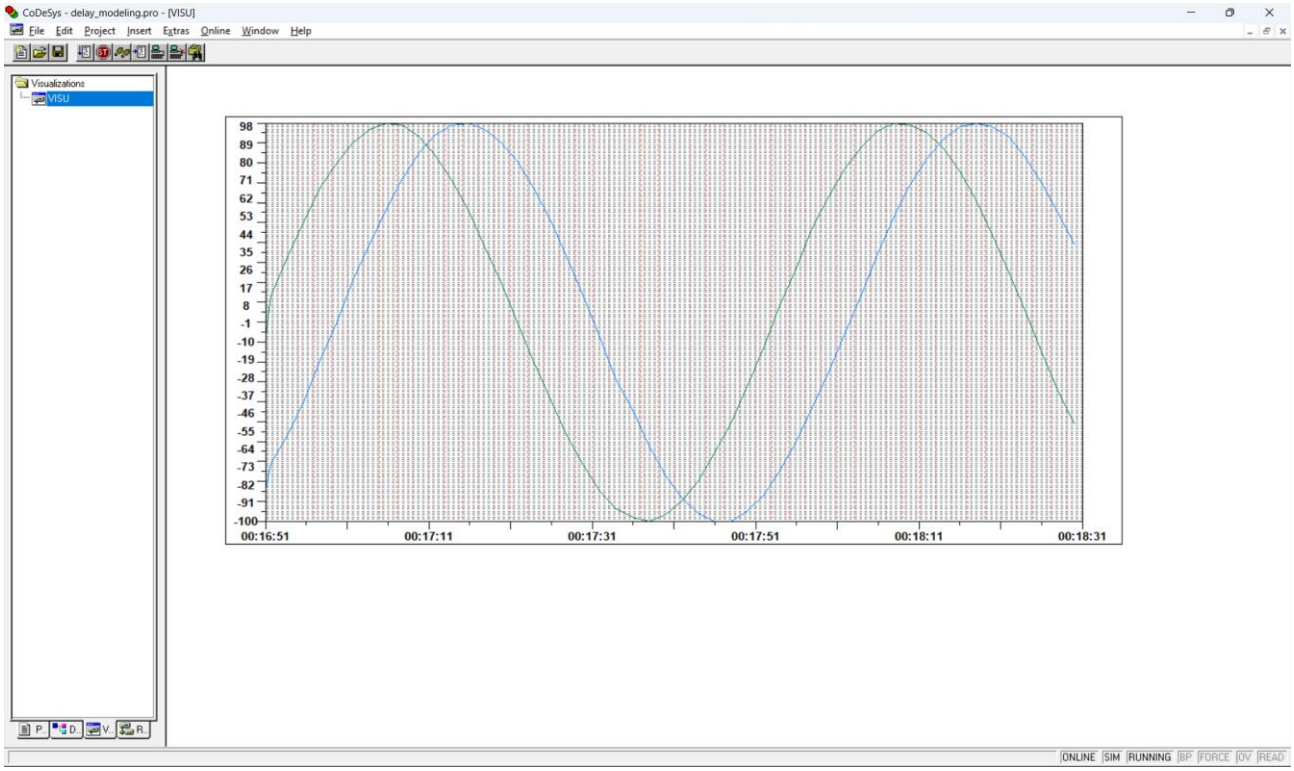


Рисунок 5.5 - Графіки заданого значення та із врахуванням інерційності.

Контрольні запитання:

1. Які сигнали може формувати функціональний блок GEN?
2. Який тип сигналу використовується в нашому прикладі?
3. Час запізнювання в нашому прикладі?
4. Навіщо використовуються стробуючі імпульси?
5. Який оператор циклу використовується в даному прикладі?

Комп'ютерний практикум №6. Візуалізація технологічного процесу із підвищеними вимогами до зручності інтерфейсу

Тривалість заняття – 2 години.

Системи керування технологічним процесами видобутку водню термохімічними та електрохімічними методами завжди були пов’язані із обробкою великої кількості аналогових та дискретних змінних.

Для стеження за протіканням складного технологічного процесу для операторів підприємств основні параметри роботи зводили на загальний пульт керування, зовнішній вигляд якого представлений на рисунку 6.1.

Рисунок 6.1 - Пульт керування атомною електростанцією.

Із розвитком комп'ютерних технологій такі стенди почали замінювати моніторами, на яких відображалась поточна інформація з технологічного процесу. При цьому, оскільки кількість інформації на моніторі значно обмежена, у порівнянні із пультом, представленому на рисунку 6.1, проектувальнику доводилось виводити лише найбільш важливі з його точки зору параметри, а інші давати в доступ операторові лише по вході в налаштування конкретного об’єкту.

Ще Дмитро Павличко у своїй пісні «Два кольори» надавав відповідність певного кольору своєму функціональному призначенню «Червоне – то любов, а

чорне – то журба». Із збільшенням кількості кольорів, що відрізняються оператором між собою, в промисловості сформувалась певна відповідність кольорів їх призначенню у людино – машинному інтерфейсі. Відповідно, палітра кольорів, що використовуються для позначень певних станів системи представлена в таблиці 6.1.

Таблиця 6.1. Відповідність кольорів функціональному призначенню.

Назва кольору	Колір	Призначення
Світло-сірий		Загальний задній фон екрану
Біло-сірий		Для відображення насосів, двигунів, обладнання тощо
Сірий		Ненавігаційні кнопки, індикація вимкненого стану двигунів, обладнання
Темно-сірий		Назва змінних, лінії протікання процесу, обриси технологічних ємностей, баків
Чорний		Тексти та позначення
Темно-синій		Значення змінних процесу, вихідні сигнали регуляторів, режими роботи
Темно - зелений		Установки регуляторів та інші змінні, що вводяться оператором
Світло-блакитний		Бажані робочі діапазони змінних або умови роботи
Синьо - зелений		Лінії рівня ємностей, криві перехідних процесів
Коричневий		Криві перехідних процесів, додаткові змінні
Червоний		Найвищий рівень загрози, перший рівень пріорітету
Жовтий		Другий рівень приорітету
Помаранчевий		Третій рівень пріорітету
Пурпуровий		Четвертий рівень пріорітету для діагностики

Кольори «червоний», «жовтий», «помаранчевий», «пурпуровий» повинні використовуватись лише для нештатних ситуацій, для інших цілей їх використовувати заборонено.

Для заднього фону необхідно використовувати сірі кольори, оскільки вони найкраще заглушують відблиски на екрані, при цьому ж забезпечують якісне освітлення місця оператора.

Здійснимо реалізацію візуалізації технологічного процесу для задачі відслідковування рівня в ємності з функцією візуалізації попередження та аварії. Для цього система візуалізації повинна містити наступний функціонал.

1. Завдання рівня здійснювати від повзунка у відсотках.
2. Додатково візуалізацію рівня відображати за допомогою стрілки «вимірювача».
3. Повинні відображатися значення завдання і поточного рівня.
4. Попередження та аварію має відображатися в полі відображення рівня.
5. Кольори повідомлень повинні відповідати стандартам високоефективного ЛМІ (людинно – машинного інтерфейсу).

1. Заповнення фону відображення. Для цього в основному вікні візуалізації натискаємо праву клавішу і далі вибираємо Select background bitmap, як це представлено на рисунку 6.2.

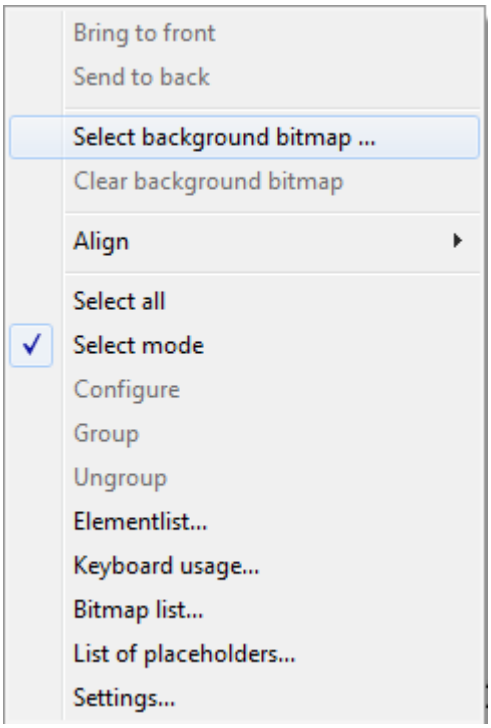


Рисунок 6.2 – Вікно налаштування фону відображення.

Далі вибираємо файл світло-сірого фону, що забезпечує незначне відсвічування екрану оператора, що відповідає вимогам високоефективного ЛМІ. При цьому фон зміниться, як це представлено на рисунку 6.3.

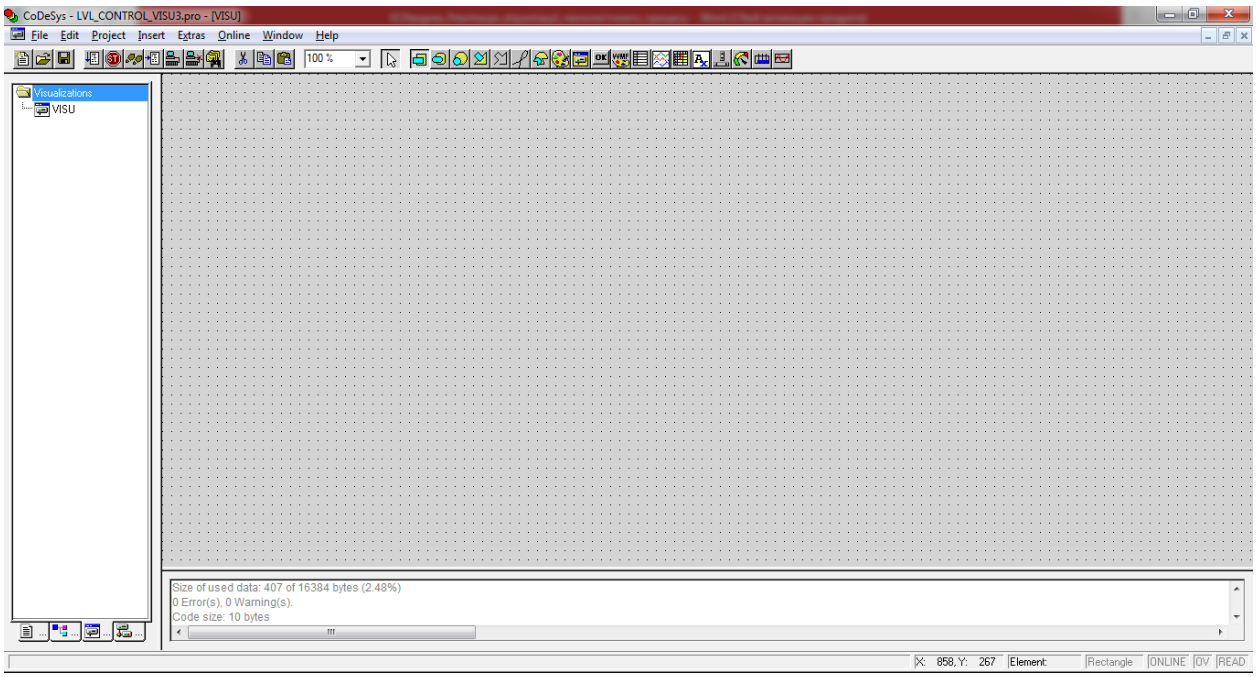


Рисунок 6.3 – Вікно візуалізації із сірим фоном.

Далі конфігуруємо вимірювач (meter) для візуалізації завдання рівня у відсотках, як це представлено на рисунку 6.4.

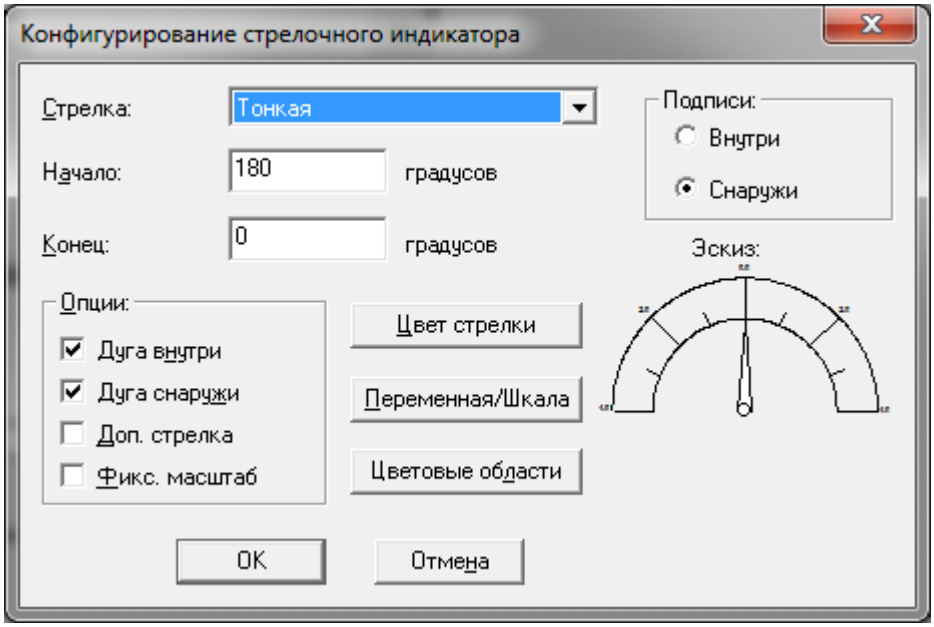


Рисунок 6.4 - Загальне вікно налаштування вимірювача.

Для початку заходимо у вкладку змінна/шкала PLC_PRG.Y1 для встановлення завдання у відсотках і виставлення градуювання різних станів роботи системи, як це представлено на рисунку 6.5.

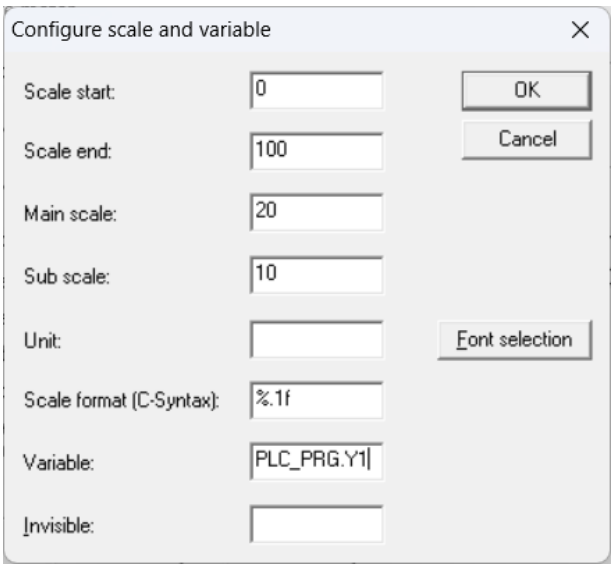


Рисунок 6.5 - Налаштування масштабу сигналу у відсотках.

Далі вибираємо області кольору, при цьому, розділяємо їх на наступні:
- 0..60% нормальна робота (колір нормальної роботи);
- 60..80% - попередження (інший колір за тоном);
- 80..100% - аварія (інший колір за підсвічуванням).
Відображення зони дії кольорів представлене на рисунку 6.6.

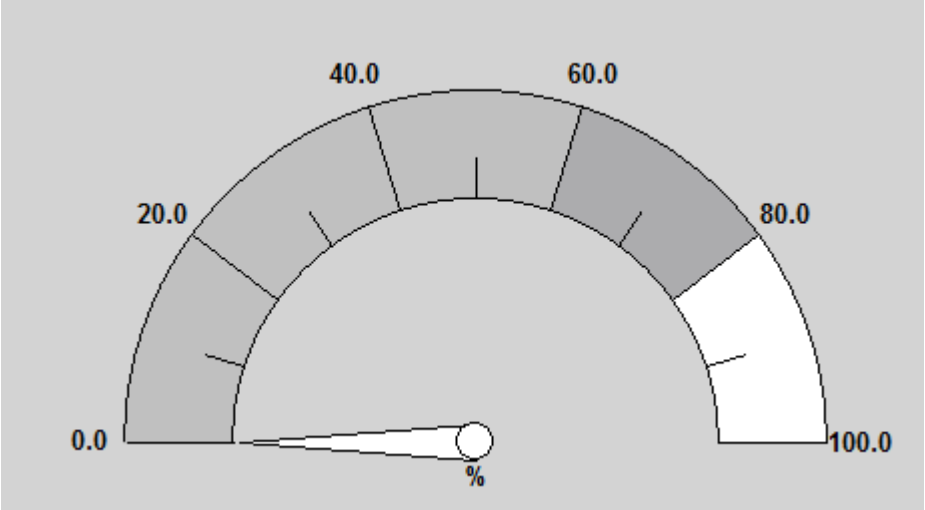


Рисунок 6.6 - Градуювання шкали відображення заповнення у відсотках.

Білий колір привертає увагу, але не є основною для формування повідомлення для оператора.

Далі сформуємо задатчик поточного рівня від повзунка Scrollbar, як це представлено на рисунку 6.7. При цьому налаштування повзунка матиме вигляд, де PLC_PRG.X1 є вхідною змінною для програми PLC_PRG.

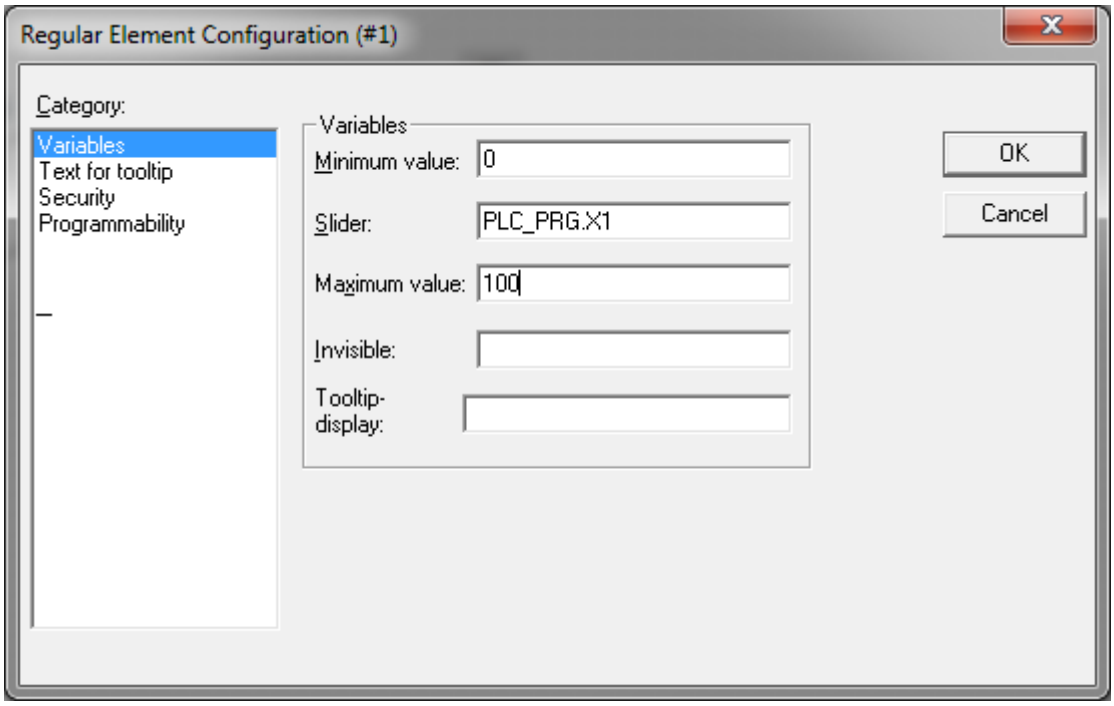


Рисунок 6.7 – Налаштування повзунка.

Частина програми, що відповідає роботі Scrollbar та Meter має вигляд, представлений на рисунку 6.8.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     X1: REAL;
0004     Y1: REAL;
0005 END_VAR
0006
```

```
0001 X1;
0002 Y1;
0003 Y1:=X1;
0004
```

Рисунок 6.8 – Програма роботи блоків відображення.

При тестуванні програми зміни положення повзунка, маємо зовнішній вигляд екрану візуалізації, представлений на рисунку 6.9.

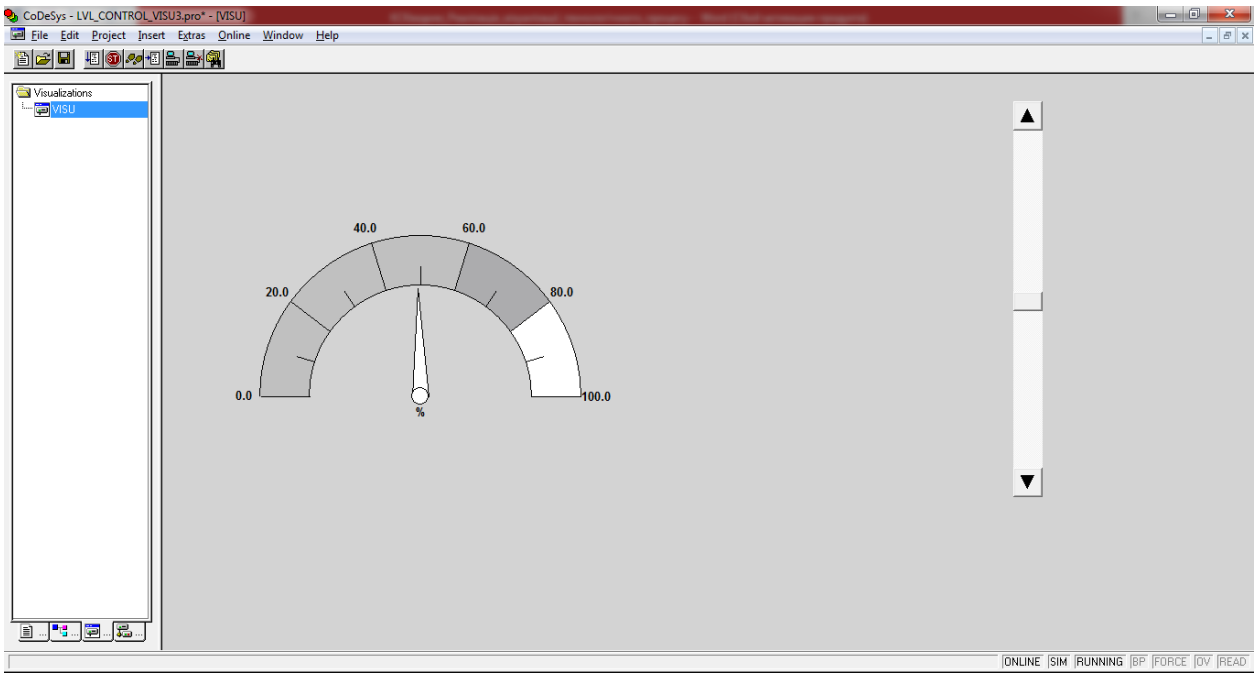


Рисунок 6.9 – Екран візуалізації із блоками повзунка та вимірювача.

Далі вводимо резервуар відображення рівня, що можна побачити на рисунку 6.10.

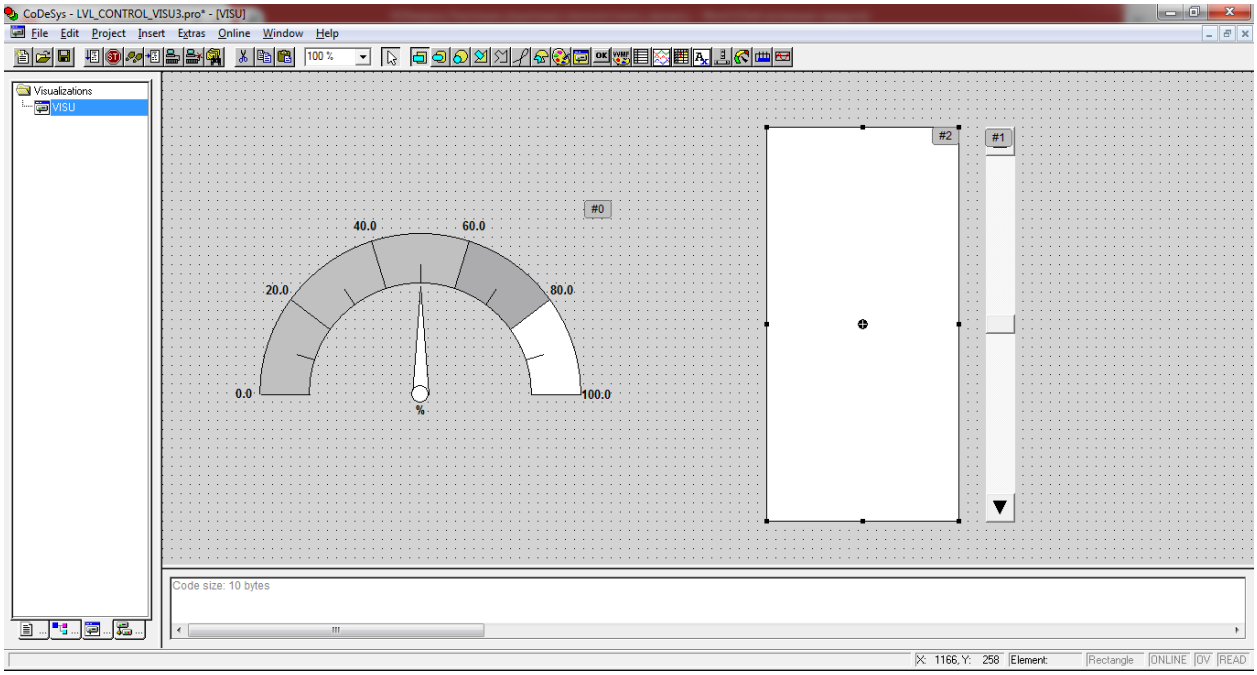


Рисунок 6.10 – Візуалізація із блоком відображення резервуару.

Налаштування блоку відповідають кольору для контуру, що вибирається з палітри доступних кольорів, представлених на рисунку 6.11.

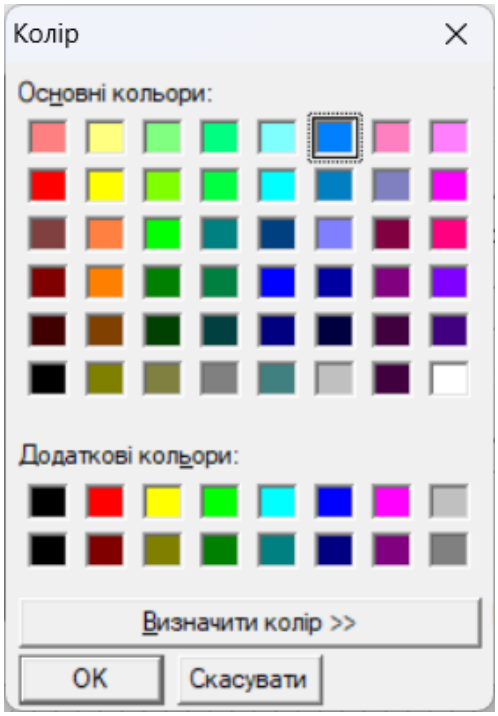


Рисунок 6.11 – Палітра і вибір кольору відображення рівня.

Крім того, налаштовується товщина ліній, як це представлено на рисунку 6.12.

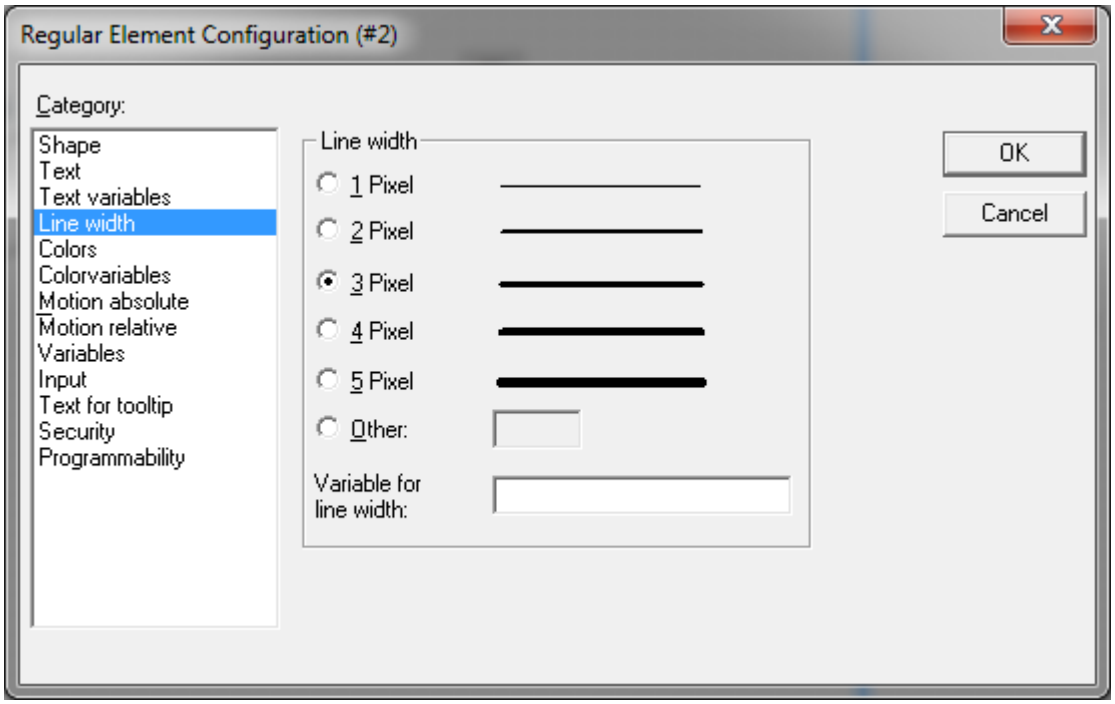


Рисунок 6.12 – Налаштування товщини лінії відображення рівня.

Налаштування вертикальної вісі представлені на рисунку 6.13.

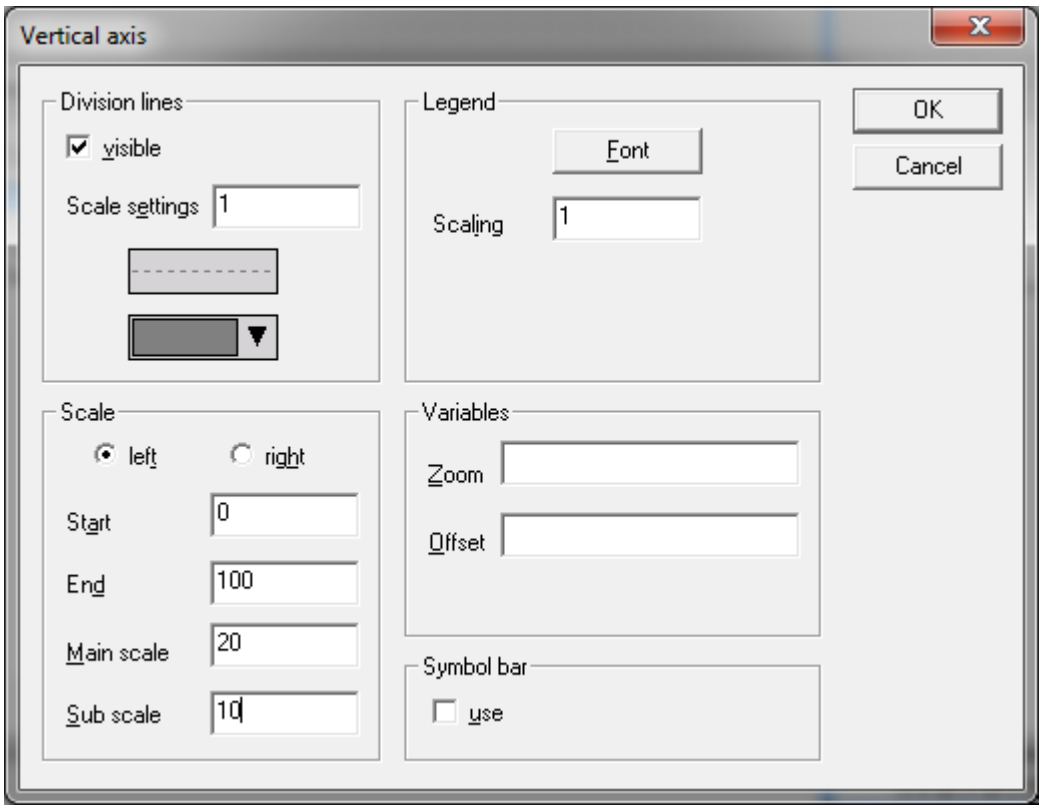


Рисунок 6.13 – Налаштування масштабів вертикальної вісі.

Далі запишемо програму формування повідомлень про нештатні ситуації з визначенням попереджень і аварійних ситуацій.

```
(*Визначення змінних*)
PROGRAM PLC_PRG
VAR
  Y1: REAL;
  X1: REAL;
  WARNING: BOOL := TRUE;
  ALARM: BOOL := TRUE;
  WARNING_GRAPH: REAL;
  ALARM_GRAPH: REAL;
  GEN1: GEN;
  AL_BLINK: BOOL;
END_VAR
(*Основна частина програми*)
Y1:=X1;
IF 60<Y1 AND Y1<80 THEN WARNING:=FALSE; ALARM:=TRUE;
ELSIF Y1>=80 THEN WARNING:=TRUE; ALARM:=FALSE;
```

```
ELSE WARNING:=TRUE; ALARM := TRUE;
END_IF;
WARNING_GRAPH:=BOOL_TO_REAL(NOT(WARNING) AND ALARM)*60;
ALARM_GRAPH:=BOOL_TO_REAL(NOT(ALARM) AND WARNING)*80;
GEN1(MODE:=TRIANGLE_POS, BASE:=TRUE, PERIOD:=t#1s ,CYCLES:= ,AMPLITUDE:=5 , RESET:=ALARM, OUT=>); (*Генератор мерехтіння*)
IF GEN1.OUT>2 THEN AL_BLINK:=TRUE;
ELSE AL_BLINK:=FALSE;
END_IF;
```

Після запуску програми на екрані відображається поточне і задане значення рівня, як це представлене на рисунку 6.14.

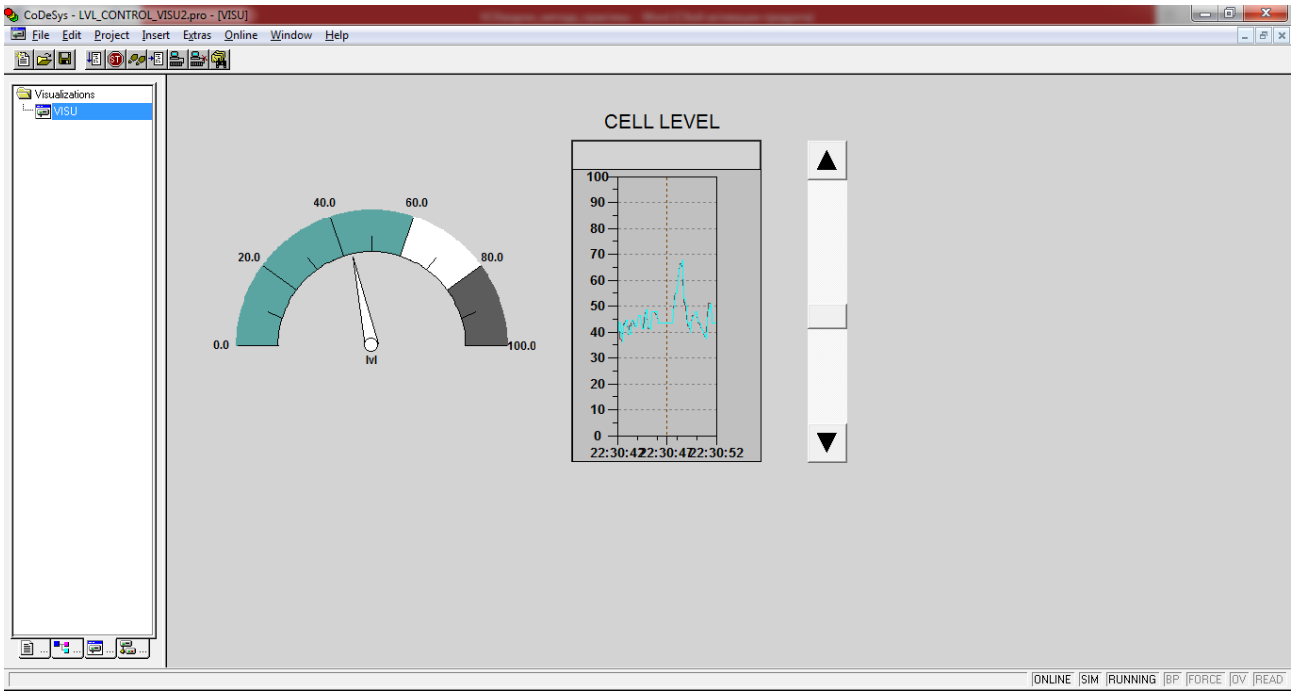


Рисунок 6.14 - Кінцевий формат візуалізації завдання і поточного рівня електроліту згідно високоефективного людино – машинного інтерфейсу.

Контрольні запитання:

1. Назвіть тривожні кольори для візуалізації.
2. Яким кольором відображається рівень рідини?
3. Які кольори доцільно вибирати для фону зображення?
4. Який колір вибирається для текстових повідомлень?
5. Назвіть пріоритетність кольорів аварій та попереджень.

Комп'ютерний практикум №7. Система керування електролізом на базі програмованого логічного контролера

Тривалість заняття – 2 години.

Керування електролізом здійснюється двома шляхами або їх комбінацією:

- шляхом керування провідністю речовини;
- шляхом керування рівнем електроліту в електролізері.

Керування провідністю здійснюється ПІД – регулятором, що впливає на перетворювач постійного струму. В ряді випадків для спрощення налаштування використовується простий П – регулятор.

Керування рівнем здійснюється за допомогою електромагнітного клапану, що подає електроліт в електролізер. При цьому відповідно до типу датчика може використовуватись як аналоговий датчик рівня, так і два дискретних. Оскільки керування за двома дискретними датчиками досліджується в інших дисциплінах, присвячених програмуванню контролерів, здійснюємо керування за допомогою аналогового датчика рівня. При цьому регулювання рівня можливе за допомогою позиційного регулятора.

Нагадуємо, що позиційний регулятор – це релейний регулятор, що приймає лише два стани – вимкнено та ввімкнено, тобто має дві позиції вихідного елементу.

При використанні комбінації керувань будується схема керування із незалежним керуванням контурами. При цьому слід відзначити, що одночасне регулювання контурами із єдиним тактом при цьому не рекомендоване, бо є складним з точки зору налаштування, як і будь-яка система MIMO (MANY INPUTS MANY OUTPUTS). Іншою особливістю є те, що ПІД – регулятор в цьому випадку повинен бути із фіксованим циклом, тобто FIXEDCYCLE. Проте, знову, таки для спрощення для керування струмом або провідністю використовуватимемо простий П – регулятор.

Розглянемо приклад реалізації такої системи із використанням виклику внутрішніх програм.

За замовчуванням під час початку циклу в контролері виконується програма типу PLC_PRG. Виберемо мову програмування CFC для зручності відображення блоків регулювання.

Для виклику іншої програми до неї треба прописати звернення у основній програмі. Ця функція реалізовується за допомогою генератора імпульсів BLINK, що запускається при вмиканні живлення контролера (знаходиться в бібліотеці Util.lib). При цьому система керування провідністю працюватиме не постійно, а лише 50% часу, щоб не конфліктувати із впливом системи керування рівнем. Час роботи і час паузи встановлюємо рівними, наприклад, п’яти секундам.

Блок HYSTERESIS підключаємо з бібліотеки Util.lib з розділу Analog monitors. При цьому на вимірювальний вхід контролера подаємо сигнал з першого аналогового входу контролера. Особливістю блоку Hysteresis є те, що він працює із цілочисельними даними типу REAL, тобто необхідно встановити блок перетворення форматів REAL_TO_INT. В цьому разі основна програма приймає вигляд, представлений на рисунку 7.1.

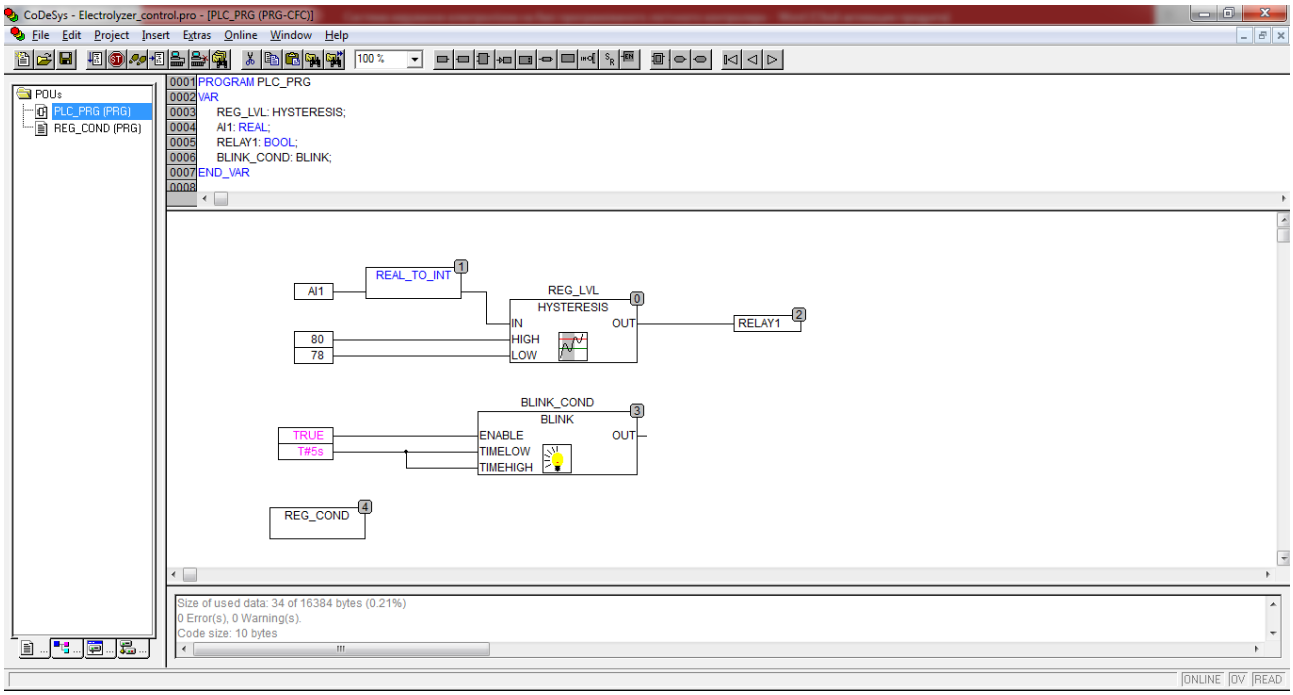


Рисунок 7.1 - Основна програма роботи керування регулюванням температури.

Далі складаємо підпрограму пропорційного керування провідністю за допомогою аналогового сигналу, що подається на напівпровідниковий регулятор потужності.

Значення AO1 є вихідним сигналом керування потужністю, причому воно формується в форматі WORD, оскільки потужність не може бути від'ємним значенням:

AO1:=REAL_TO_WORD(MAX(0,ENABLE*Kreg*(UST-COND)));

де UST – усталене значення провідності,

COND – поточне значення провідності.

Коефіцієнт підсилення регулятора має бути встановлений емпіричним чином, в залежності від вимог до динамічних характеристик системи. Він повинен бути ненульовим, що вказується за допомогою асистента вводу в

початкових умовах. Приклад асистента вводу для вводу коефіцієнта регулятора представлений на рисунку 7.2.

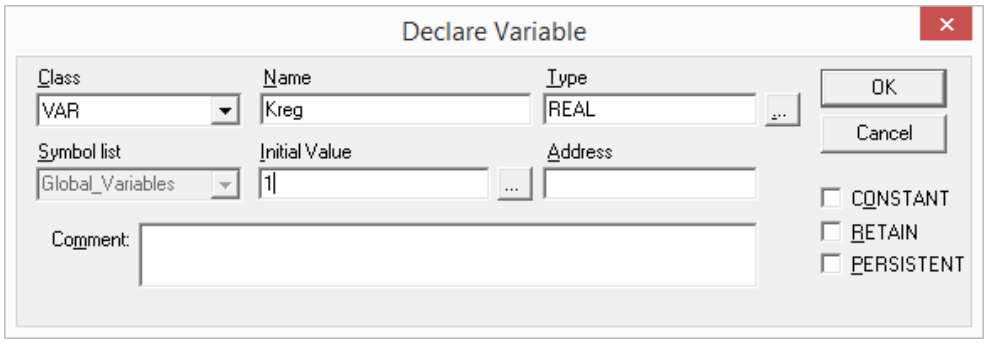


Рисунок 7.2 - Приклад асистента вводу ненульових початкових значень.

Для формування команди активації П – регулятора використовуємо сигнал ENABLE із виходу генератора імпульсів основної програми:

ENABLE:=BOOL_TO_REAL(PLC_PRG.BLINK_COND.OUT);

При цьому підпрограма керування провідністю матиме вигляд, представлений на рисунку 7.3.

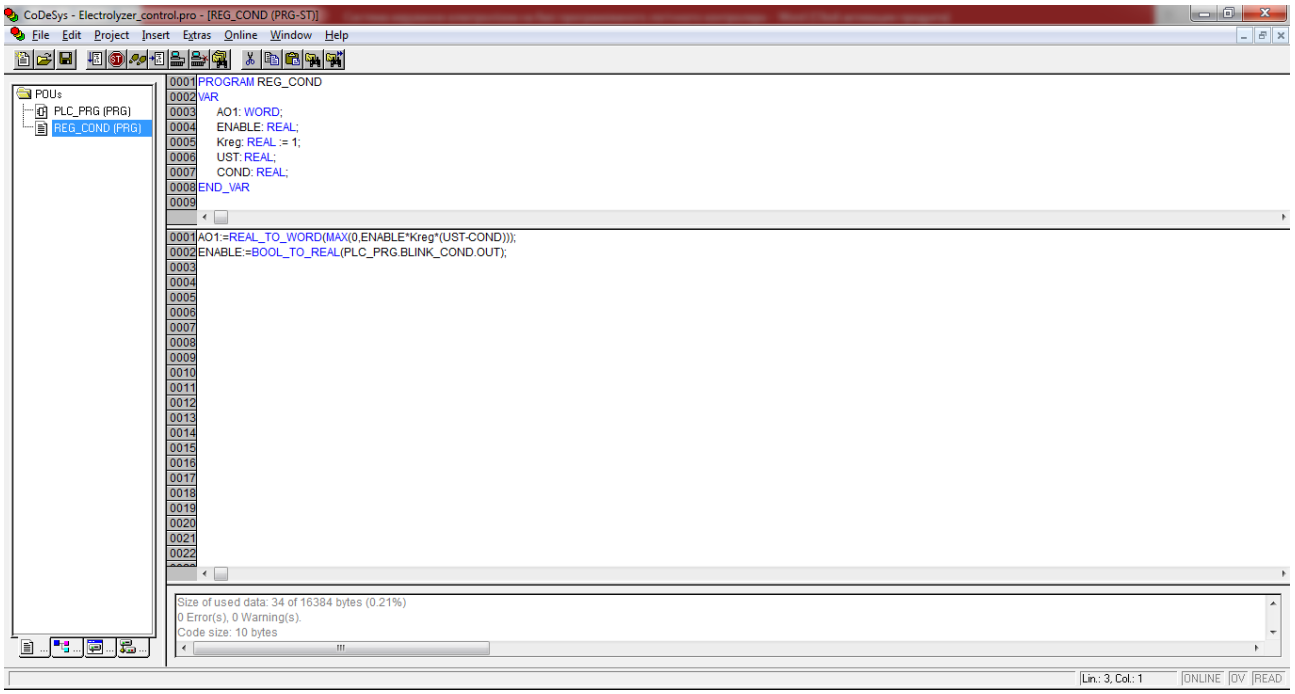


Рисунок 7.3 - Програма пропорційного керування потужністю електролізу.

Для тестування працездатності алгоритму в цілому, доцільно використовувати можливості візуалізації та емуляції технологічного процесу, аналогічно попередній задачі.

Контрольні запитання:

1. Що таке П – регулятор?
2. Назвіть переваги П – регулятора?
3. Що таке позиційний регулятор і для чого він використовується?
4. Для чого і як налаштовується HYSTERESYS?
5. Що таке MIMO?

Комп'ютерний практикум №8. Робота із цільовими файлами

Тривалість заняття – 2 години.

Програмне середовище Codesys може бути використане для різних програмованих логічних контролерів. Особливістю можливостей Codesys є наявність в контролері спеціалізованої система виконання (Control Runtime System), що встановлюється в контролер при його виготовленні. В той час як робота із програмними блоками POU та вузлами візуалізації є спільним для всіх виробників, то налаштування входів і виходів контролера має суттєві відмінності. Для цього використовуються так звані цільові файли прив'язки, що надаються виробниками контролерів, як на безоплатній, так і на платній основі. В якості прикладу налаштування цільового файлу візьмемо контролер ПЛК100.Р-L із дискретними входами і виходами від українського виробника «Акутек».

Щоб створити новий проект, потрібно вибрати Файл/Створити або скористатися однойменною кнопкою на панелі інструментів. Після створення проекту потрібно вибрати цільовий файл, який відповідає імені контролера. Цільовий файл повинен бути встановлений заздалегідь. Вікно вибору цільового файлу показано на рисунку 8.1.

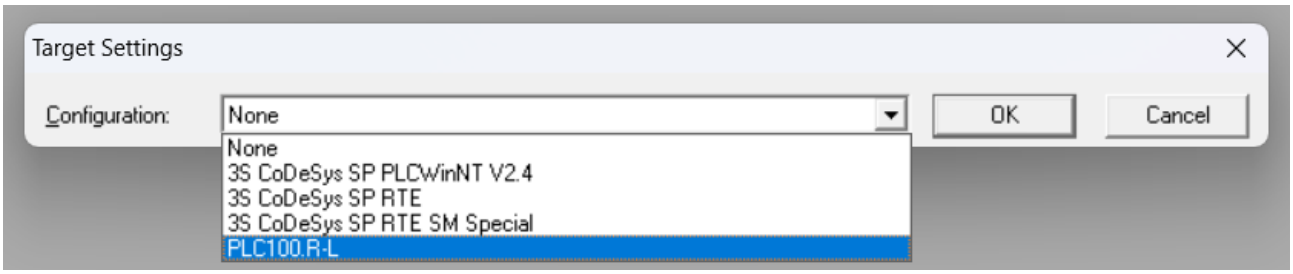


Рисунок 8.1 – Вікно вибору цільового файлу

Після цього відкриється вікно налаштувань цільового файлу, представлене на рисунку 8.2. Як правило, параметри встановлюються виробником і не потребують зміни (за винятком зміни обсягу Keep memory).

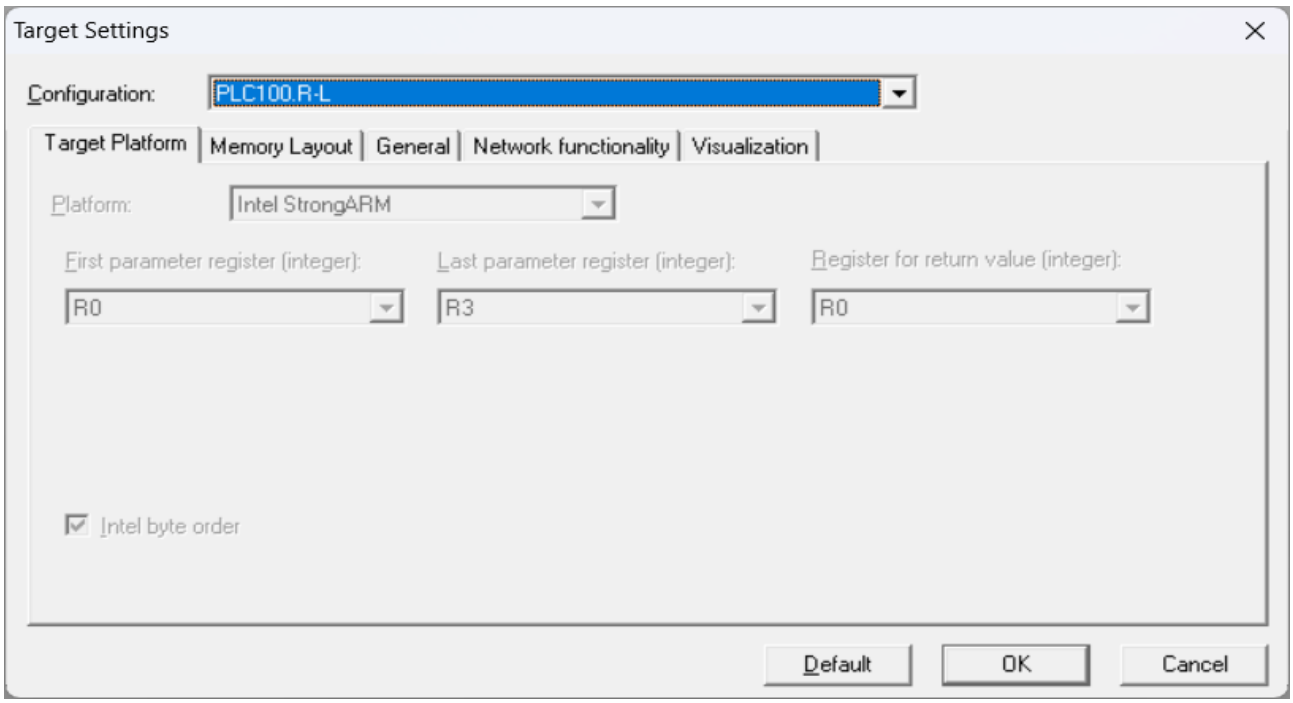


Рисунок 8.2 – Налаштування області памяті входів/виходів контролера.

Після підтвердження налаштувань цільового файлу потрібно створити основний програмний блок POU (головну програму проекту). Вікно цього діалогу показано на рисунку 8.3. Головна програма завжди повинна мати тип PROGRAM і назву PLC_PRG. Тому в цьому діалоговому вікні потрібно вибрати тільки мову блоку POU, в нашому випадку це буде текстова мова програмування ST.

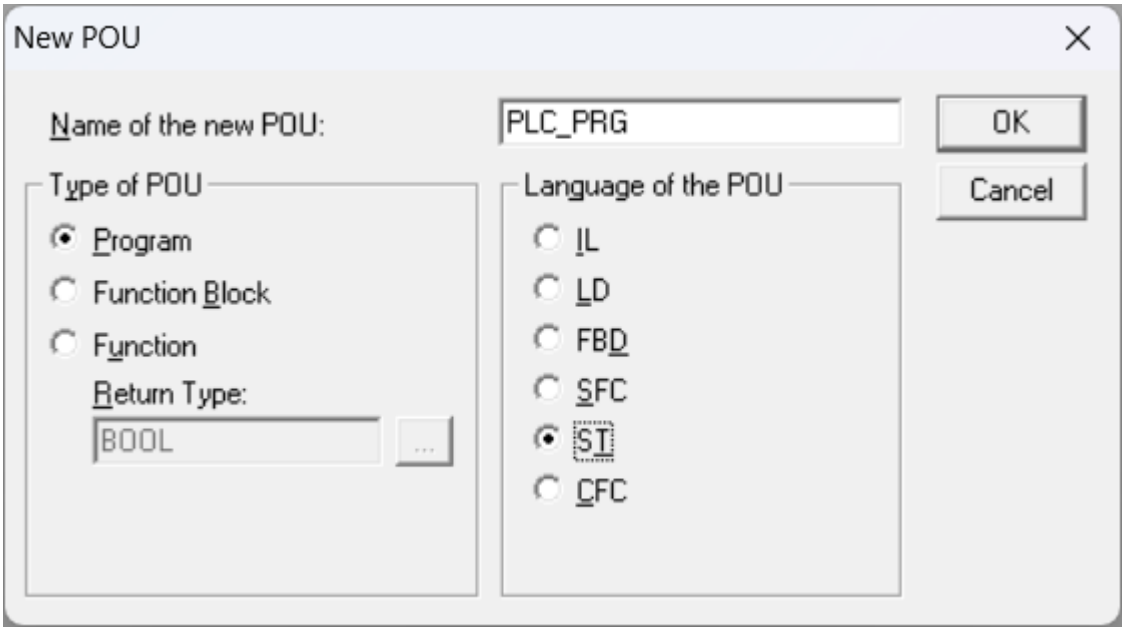


Рисунок 8.3 – Вікно створення основного POU

Далі привязка до дискретних входів та виходів контролера здійснюється в блоці PLC_configuration у вкладці Resources, як це представлено на рисунку 8.4.

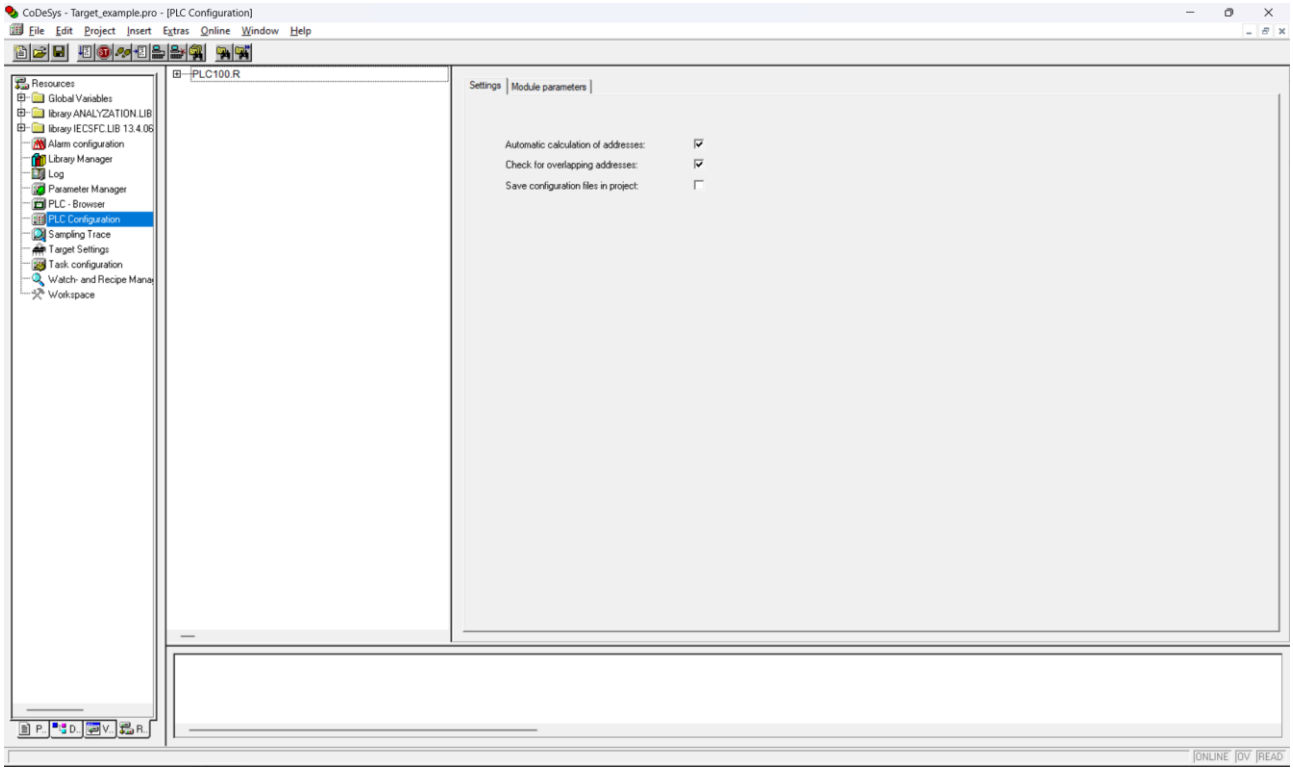


Рисунок 8.4 – Вікно конфігурування ПЛК.

Відкриваємо конфігурацію самого контролера натисканням «+», отримуємо повну конфігурацію вхідних та вихідних сигналів контролера ПЛК100.P-L, представлену на рисунку 8.5.

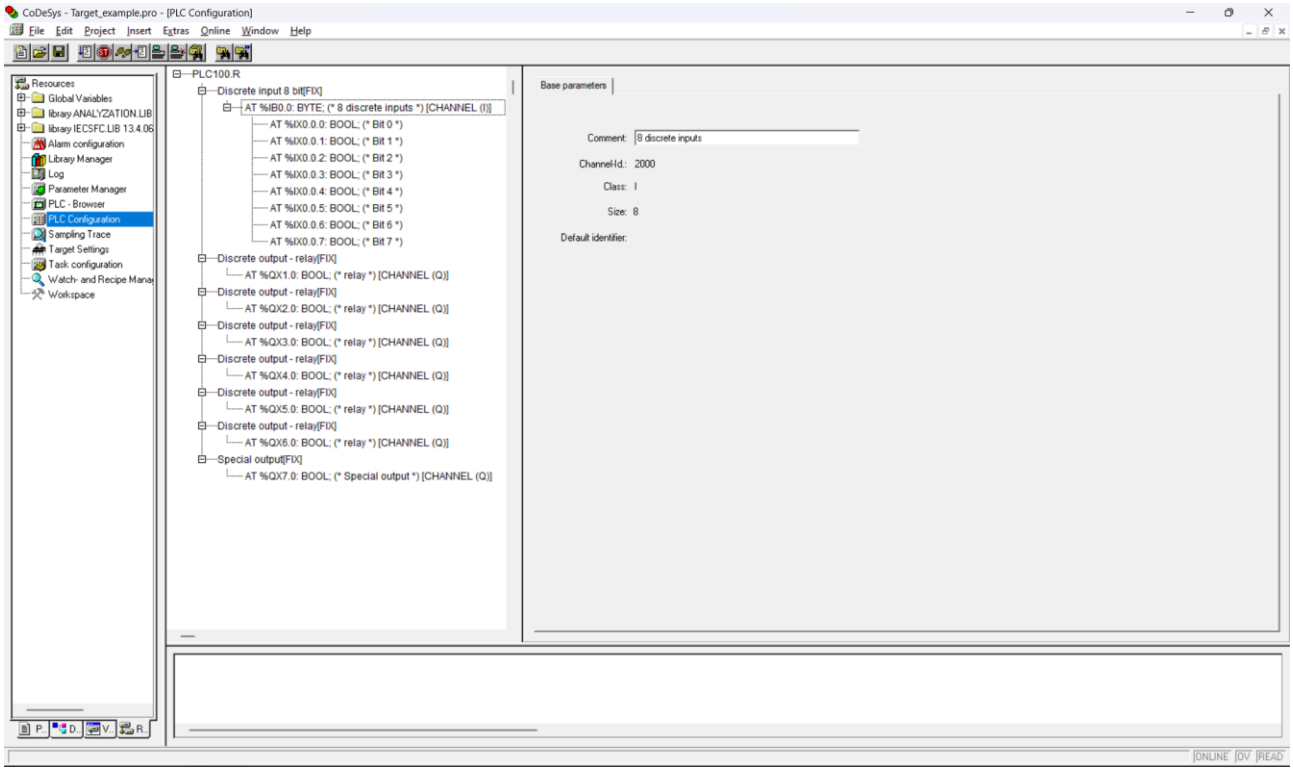


Рисунок 8.5 – Повна конфігурація вхідних та вихідних сигналів ПЛК100.

Далі необхідно прив’язати апаратні канали до змінних в програмі контролера. Найпростішим варіантом є безпосереднє позначення змінної в конфігурації ПЛК. Лівим кліком миші на початку назви поля адреси відкриваємо поле налаштування назви змінної, як показано на рисунку 8.6 для входу №8 контролера.

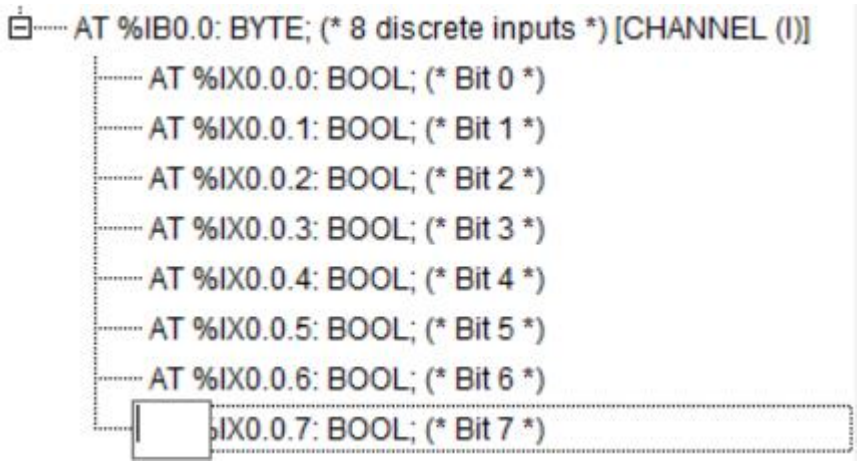


Рисунок 8.6 – Поле формування назви змінних

Відповідно, для вхідних і вихідних змінних присвоюємо назви з урахуванням вимог Codesys, а саме:

- в назві змінної використовується лише латиниця;
- назва змінної не може починатися із цифри;
- в якості назви не може використовуватися зарезервована команда або позначення, наприклад, IF, BOOL, END.

Приклад конфігурації для простої схеми ручного керування рівнем електроліту представлений на рисунку 8.7.

При цьому обов’язково необхідно у вікні Login поставити галочку «Режим симуляції», як це показано на рисунку 8.8, в іншому випадку під час запуску програми буде формуватися повідомлення про помилку з’єднання з контролером.

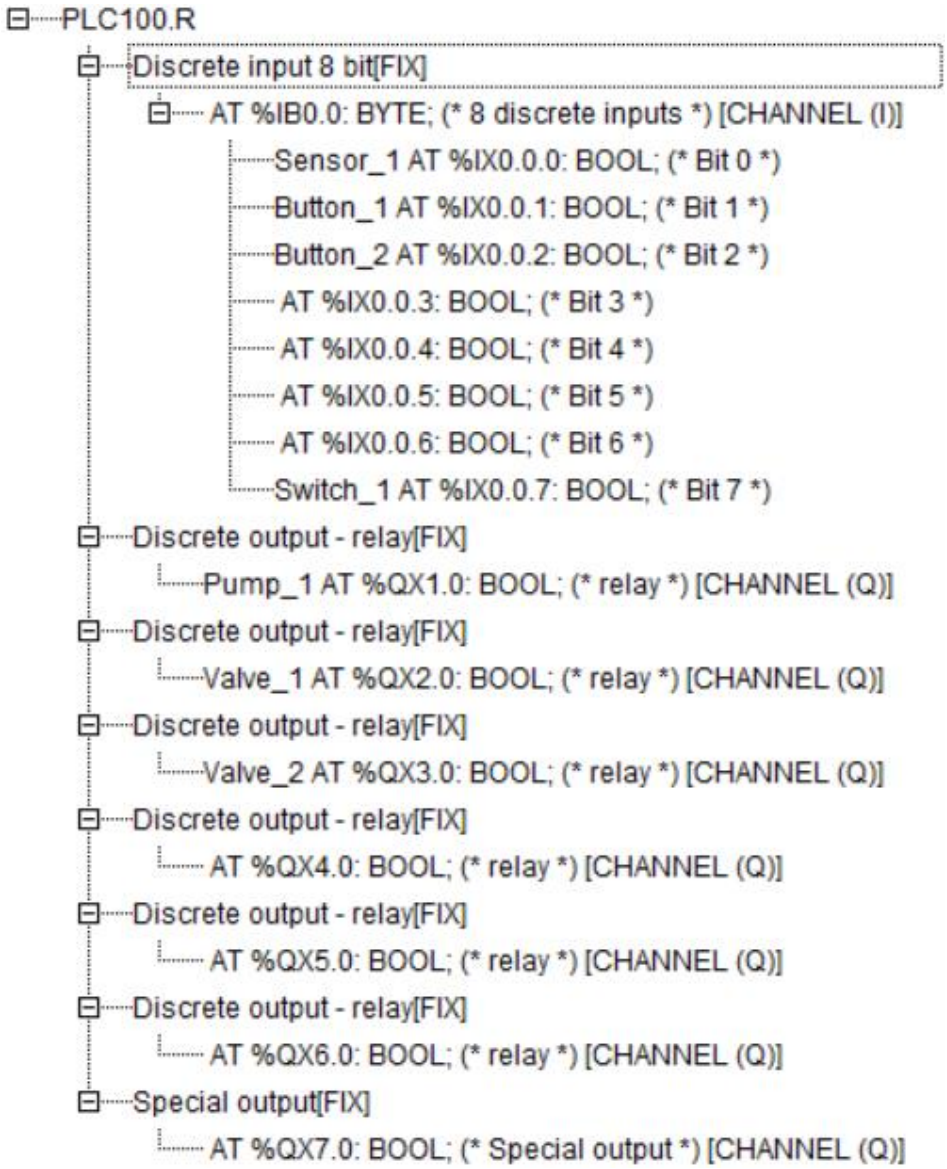


Рисунок 8.7 – Конфігурація ПЛК для ручного керування рівнем електроліту.

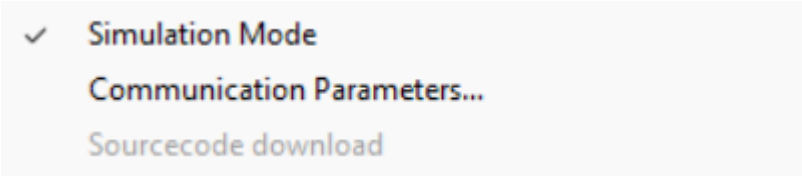


Рисунок 8.8 – Вікно налаштування в режимі симуляції.

Напишемо просту програму ручного керування рівнем.

Система керування рівнем здійснює управління насосом перекачування та клапанами подачі електроліту в комірки електролізеру №1 і №2.

Вмиканням тумблера із фіксацією (Switch_1) і наявності сигналу датчика сухого ходу (Sensor_1) система переходить в робочий режим, при цьому починає безперервно працювати насос (Pump_1). Далі кнопками без фіксації (Button_1 і Button_2) вмикаються відповідно клапани №1 і №2 (Valve_1 і Valve_2). По відключенні датчика сухого ходу або тумблера насос вимикається, керування клапанами блокується.

Основний текст програми при цьому запишеться таким чином:

```
Pump_1:=Switch_1 AND Sensor_1;
Valve_1:=Button_1 AND Pump_1;
Valve_2:=Button_2 AND Pump_1;
```

Слід відзначити, що всі змінні цієї програми вже визначені в конфігурації ПЛК і не потребують додаткового позначення. Крім того, вони є глобальними і їх імена будуть визначені для всіх POU.

Перевагою такого підходу є можливість прямого керування входами і відслідковування виходів без процесу запису змінних (команда Write variables, CTRL+F7). Для цього достатньо у вкладці конфігурації ПЛК лівим кліком активувати квадрати позначення станів змінних. На рисунку 8.9 наведено приклад стану змінних при вмиканні другого клапану.

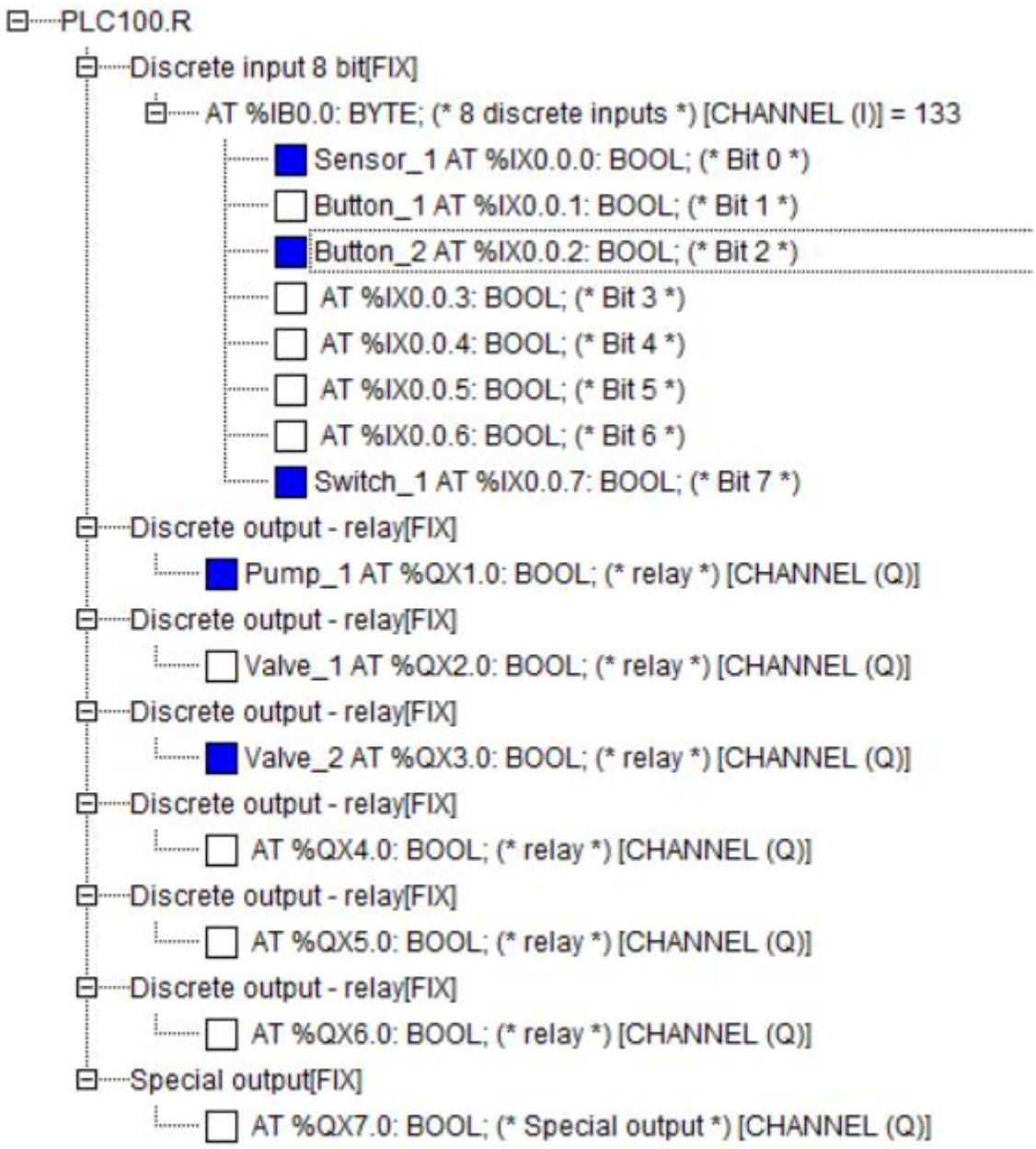


Рисунок 8.9. Стан входів/виходів ПЛК.

Контрольні запитання.

1. Що таке цільовий файл?
2. Як здійснюється прив’язка змінних ПЛК до реальних входів/виходів?
3. Як в Codesys відображається спрацювання дискретного входу?
4. Що можете сказати про змінні %IX та %QX?
5. Які типи даних використовуються в цільовому файлі для ПЛК100?

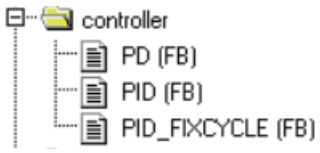
Комп'ютерний практикум №9. Синтез системи контролю технологічного параметра на основі ПІД – регулятора Codesys

Тривалість заняття – 2 години.

Використання програмованих логічних контролерів із вбудованими бібліотеками ПІД – регулювання дозволяє забезпечити точний та стабільний рівень вихідного параметру в технологічній системі. В даному прикладі ПІД-регулятор використовується для керування засувкою, яка регулює подачу електроліту до напівпровідникового перетворювача при керуванні рівня, а також теплоносія при регулюванні температури. Його завдання полягає у підтриманні заданого рівня вихідної координати, забезпеченні стабільності та швидкості реагування на зміни.

Існує декілька реалізацій ПІД – регулятора, як від самого виробника середовища Codesys 3S Software, так і від виробників окремих контролерів.

Регулятори від виробника Codesys знаходяться в бібліотеці util.lib у вкладці controller:



- PD(FB) – регулятор використовується в системах з великим ступінем астатизму;
- PID(FB) – типовий ПІД – регулятор;
- PID_FIXCYCLE(FB) - регулятор із функцією обчислення вихідного значення за зовнішньою послідовністю імпульсів.

Будемо використовувати стандартний ПІД – регулятор, представлений на рисунку 9.1, вихід якого будемо підключати на блок емуляції об'єкту. Для зручності роботи з ПІД – регулятором, як із функціональним блоком з великою кількістю входів – виходів доцільно використовувати графічну мову програмування, наприклад CFC.

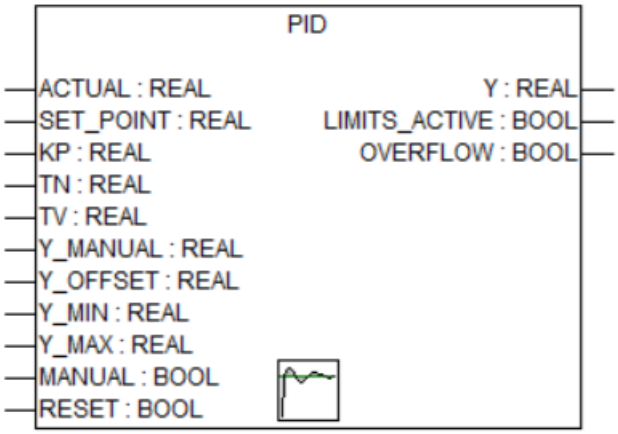


Рисунок 9.1 - Функціональний блок стандартного ПІД – регулятора від Codesys.

Недоліком цього регулятора є:

- неможливість роботи в П-, ПІ- та ПД-режимах, оскільки внутрішній код регулятора не сприймає наявність нулів для змінних KP, TN, TV;
- для нього відсутня внутрішня процедура автоматичного налаштування параметрів.

В якості керованого об'єкту використовуємо аперіодичну ланку другого порядку:

Розглянемо даний підхід на прикладі аперіодичної ланки першого порядку:

$$\frac{X_{\text{вих}}(\text{p})}{X_{\text{вх}}(\text{p})} = \frac{K}{(\text{T}_1 p + 1)(\text{T}_2 p + 1)}$$

При цьому отримуємо реалізацію даної ланки в середовищі Codesys в наступному вигляді, представленому на рисунку 9.2.

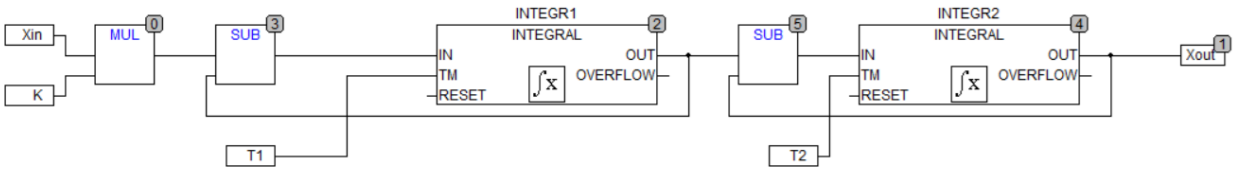


Рисунок 9.2 - Технологічний процес, що відповідає термохімічному процесу нагріву із малим чистим запізнюванням.

Виносимо реалізацію цього процесу в окремий функціональний блок. При цьому визначення змінних відбувається таким чином, що входом є тільки змінна Xin, виходом Xout, а всі інші – внутрішні:

```
FUNCTION_BLOCK Process
VAR_INPUT
  Xin: REAL;
END_VAR
VAR_OUTPUT
  Xout: REAL;
END_VAR
VAR
  K: REAL:=1;
  T1: INT:=1;
  T2: INT := 2;
  INTEGR1: INTEGRAL;
  INTEGR2: INTEGRAL;
END_VAR
```

В основній програмі, що пишемо мовою програмування ST лише робимо виклик програми керування:

```
CTRL();
```

Саму програму керування пишемо мовою CFC для зручності побудови системи регулювання із функціональними блоками. Така програма наведена на рисунку 9.3.

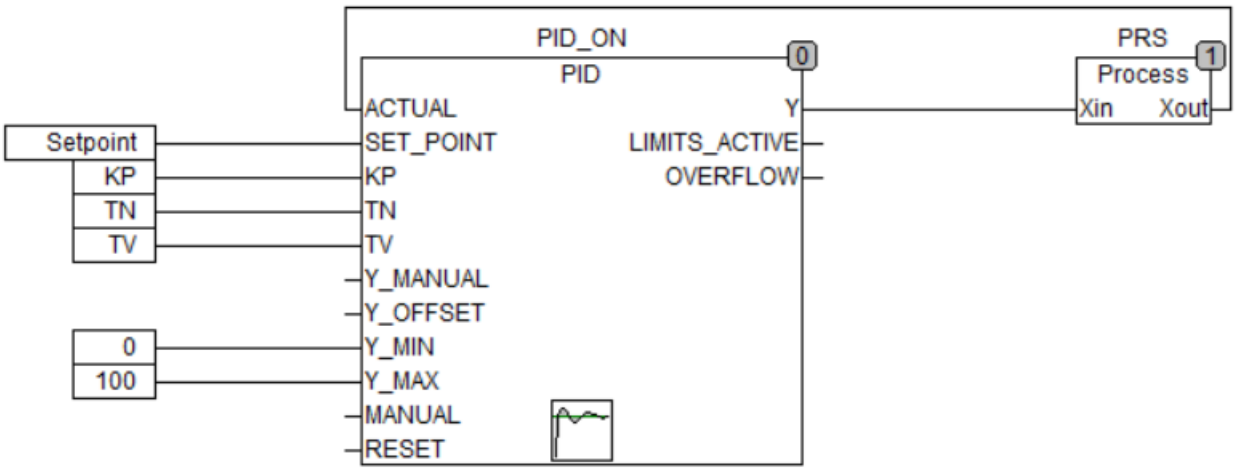


Рисунок 9.3 - Структурна схема системи керування в Codesys.

Перехідний процес за температурою є суттєво інерційним і потребує корекції параметрів регулятора, адже при заводських налаштуваннях робота системи керування є неприпустимою. Графік перехідного процесу при неналаштованому ПІД – регуляторі із усталеним значенням Setpoint:=50

представлений на рисунку 9.4. Очевидно, що в системі є суттєві коливання регульованої величини, відхилення від заданого значення перевищує 20%.

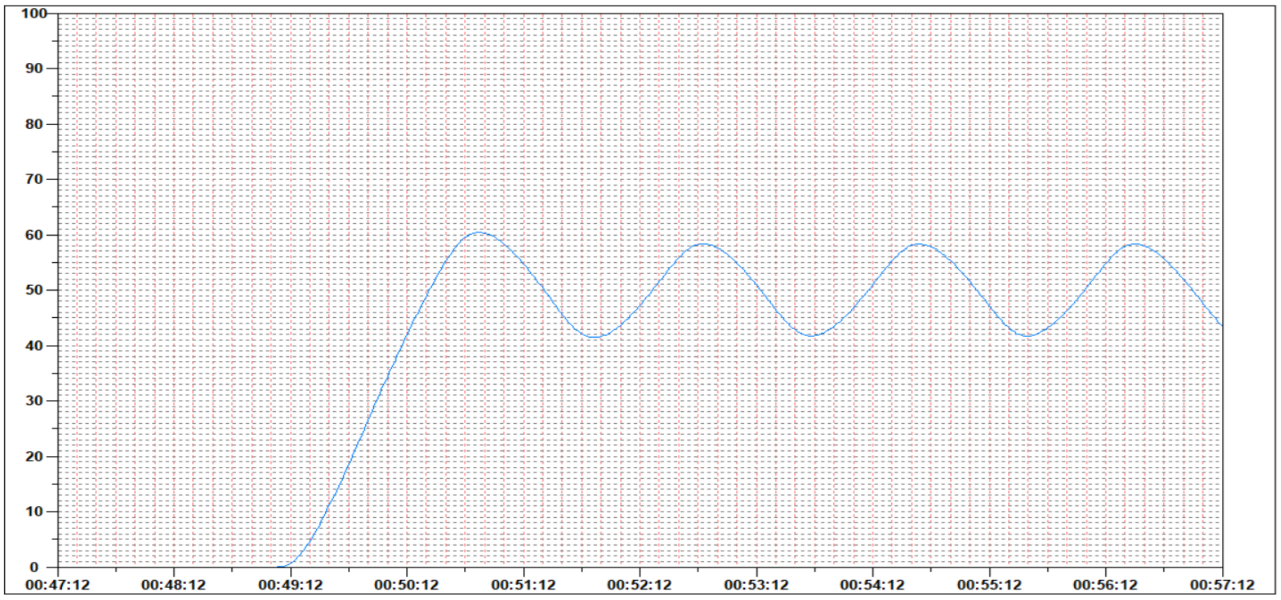


Рисунок 9.4 - Графік перехідного процесу при заводських налаштуваннях ПІД – регулятора.

Після збільшення коефіцієнту підсилення і диференціювання графік перехідного процесу приймає вигляд, представлений на рисунку 9.5. При збільшенні коефіцієнту інтегрування графік перехідного процесу приймає вигляд, представлений на рисунку 9.6.

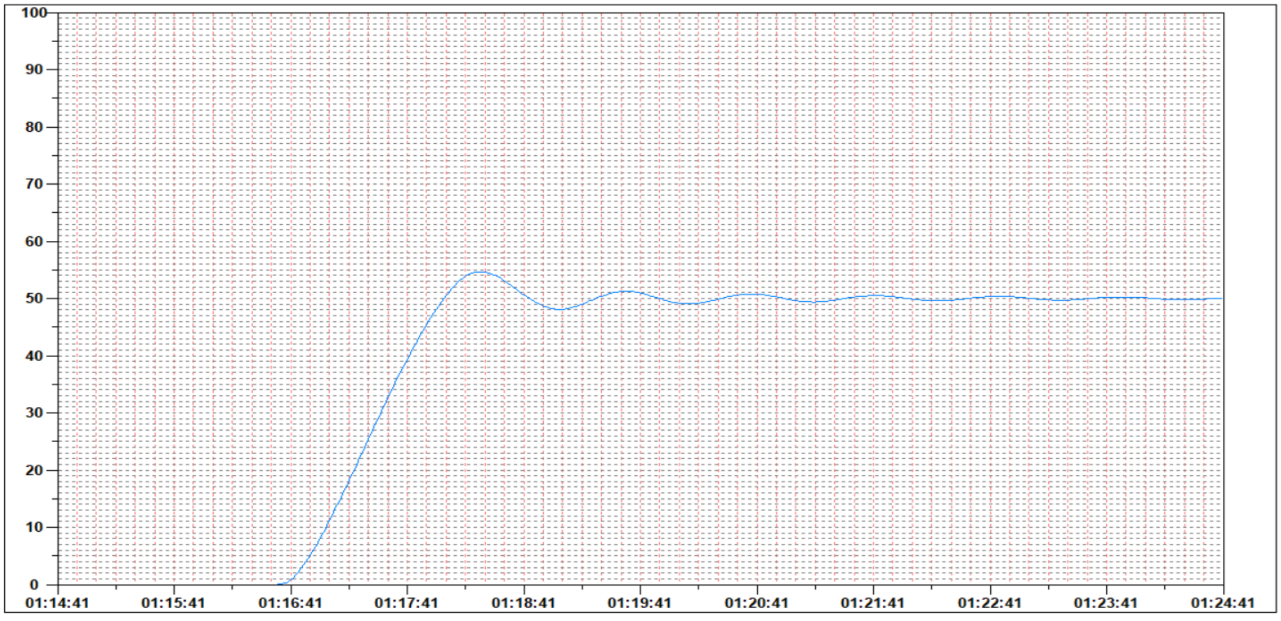


Рисунок 9.5 - Графік перехідного процесу за вихідною координатою при збільшенні пропорційної та диференційної складової.

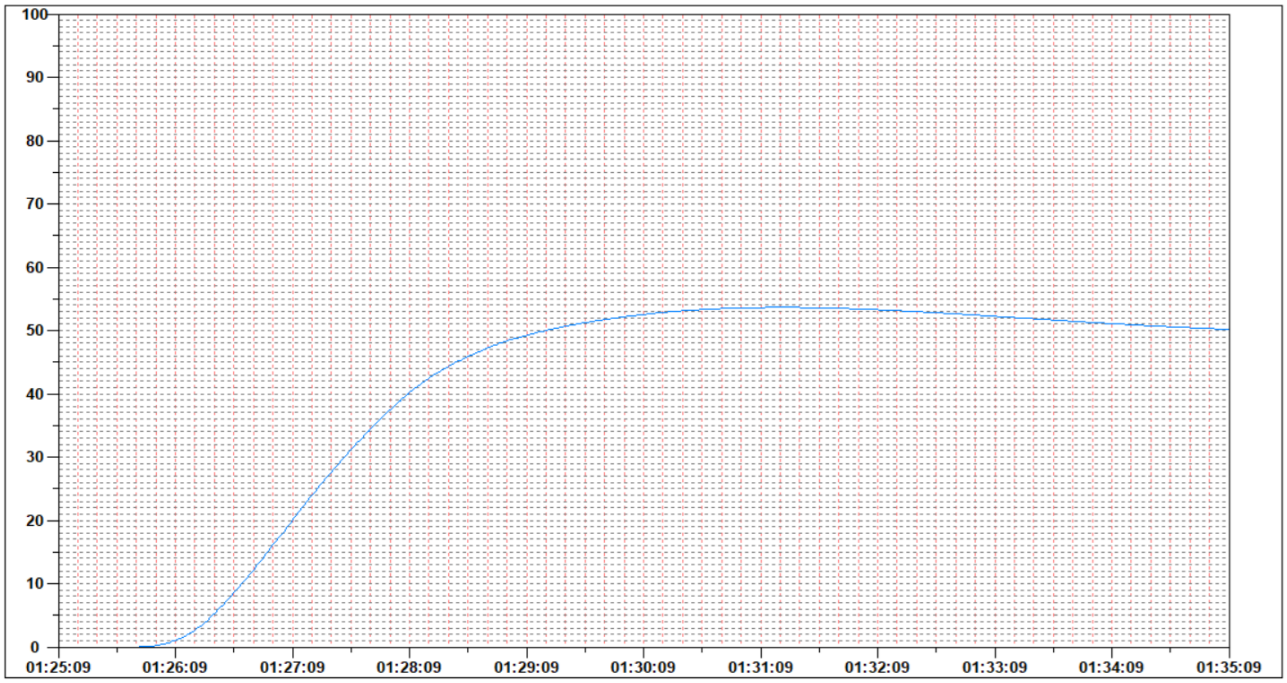


Рисунок 9.6 - Графік перехідного процесу при збільшенні інтегральної складової.

Контрольні запитання.

1. Що таке ПІД – регулятор?
2. Основні переваги ПІД – регулятора.
3. Які входи блоку ПІД – регулятора відповідають за його коефіцієнти регулювання?
4. Який характер перехідного процесу реалізований в даному прикладі?
5. Назвіть блоки ПІД – регулювання в бібліотеці util.lib і поясніть відмінності.

Перелік рекомендованої літератури:

1. Розроблення людино-машинних інтерфейсів та систем збирання даних з використанням програмних засобів SCADA/HMI. : Навч. посіб. Київ : Видавництво Ліра-К, 2020. — 594 с.
2. Пупена О. М. Розробка та використання імітаційних моделей для відлагодження програмного забезпечення програмованих логічних контролерів / О. М. Пупена, І. В. Ельперін, В. М. Кушков // Наукові праці Національного університету харчових технологій. - 2012. - № 44. - С. 6-11. - Режим доступу: http://nbuv.gov.ua/UJRN/Npnukht_2012_44_3.
3. Торопов, А. В. Комп'ютерні системи керування термохімічними та електрохімічними процесами видобутку водню. Рекомендації до виконання розрахунково-графічної роботи [Електронний ресурс] : навч. посіб. для здобувачів ступеня магістра за освітньою програмою «Інжиніринг інтелектуальних електротехнічних та мехатронних комплексів» спеціальності 141 Електроенергетика, електротехніка та електромеханіка / А. В. Торопов, А. В. Босак, Л. В. Торопова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 544.79 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2023. – 22 с. – Назва з екрана.
4. Добржанський, О. О. (2013). Застосування програмованого логічного контролеру в якості емулятора технологічного об’єкту. Східно-Європейський журнал передових технологій, 6/2 (66), 37-41.
5. Комп’ютерне управління технологічними процесами, експериментом, обладнанням. Методичні вказівки до практичних занять [Електронний ресурс] : навчальний посібник для здобувачів ступеня магістра за освітньою програмою «Інжиніринг інтелектуальних електротехнічних та мехатронних комплексів» / А. В. Торопов, А. В. Босак, Л. В. Торопова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 5,42 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022. – 90 с. – Назва з екрана.
6. Моделювання процесів і систем: комп'ютерний практикум [Електронний ресурс]: навчальний посібник для студентів освітньої програми «Інтегровані інформаційні системи» спеціальності 126 «Інформаційні системи та технології» / КПІ ім. Ігоря Сікорського; уклад. В. А. Яланецький. – Електронні текстові дані (1 файл: 1.84 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 134 с. – Назва з екрана.
7. Яланецький, В. А. Проектування інформаційних систем. Комп'ютерний практикум [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Інтегровані інформаційні системи»

спеціальності 126 «Інформаційні системи та технології» / В. А. Яланецький ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 3,13 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 138 с. – Назва з екрана.