

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДОНБАСЬКА ДЕРЖАВНА МАШИНОБУДІВНА АКАДЕМІЯ

МЕТОДИЧНІ ВКАЗІВКИ

до комп'ютерного практикуму по дисципліні
«ПРОЕКТУВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ»

Частина 1

(для студентів спеціальності 151

“Автоматизація та комп'ютерно-інтегровані технології»)

Краматорськ 2018

УДК 621.3

Методичні вказівки до комп'ютерного практикуму по дисципліні «Проектування систем автоматизації» Частина 1. (для студентів спеціальності «Автоматизація та комп'ютерно-інтегровані технології») / Укл. О. О. Сердюк, - Краматорськ: ДДМА, 2018. - 91 с.

Наведений опис інтерфейсу інструментального засобу візуального програмування контролерів CoDeSys. Освітлені особливості й приймання програмування систем автоматизації на мовах стандарту MEK 61131-3. Викладені методичні рекомендації зі створення електротехнічних схем у програмній системі EPLAN і оформленню текстової проектної документації відповідно до вимог стандартів.

Укладач СЕРДЮК Олександр Олександрович, доц.

Відп. за випуск КЛИМЕНКО Галина Петрівна, проф.

1 ВИВЧЕННЯ СИСТЕМИ ПРОГРАМУВАННЯ КОНТРОЛЕРІВ CoDeSys

Ціль лабораторної роботи: освоєння інтерфейсу й приймань роботи з інструментом програмування контролерів CoDeSys.

1.1 Головне вікно системи CoDeSys

Елементи головного вікна.

Головне вікно CoDeSys (рис. 1.1) складається з наступних елементів (у вікні вони розташовані зверху вниз):

- Меню.
- Панель інструментів, на якій перебувають кнопки для швидкого виклику команд меню.
- Організатор об'єктів, що має вкладки POU, Data types, Visualizations і Resources.
- Роздільник організатора об'єктів і робочої області CoDeSys.
- Робоча область, у якій перебуває редактор.
- Вікно повідомлень.
- Рядок статусу, що містить інформацію про поточний стан проекту.

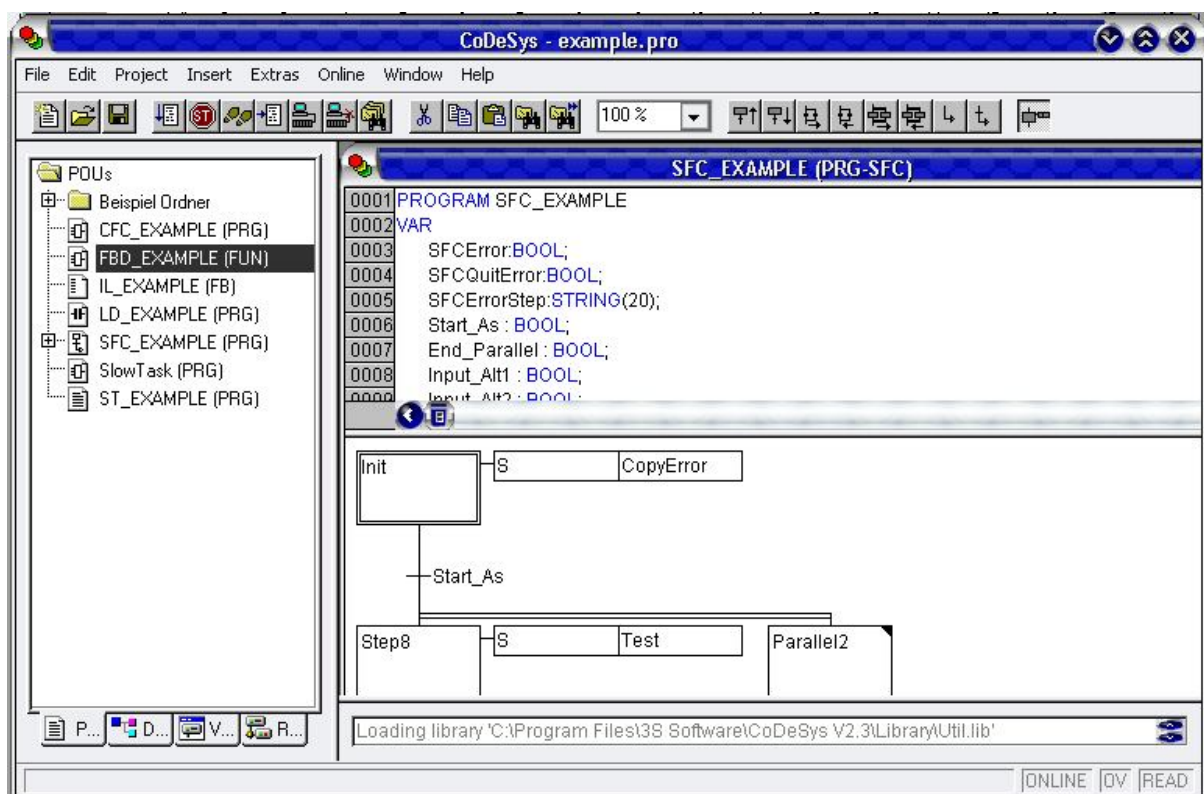


Рисунок 1.1 – Головне вікно CoDeSys

Панель інструментів, вікно повідомлень і рядок статусу не є обов'язковими елементами головного вікна.





Меню містить команди CoDeSys: File, Edit, Project, Insert, Extras, Online, Window, Help.

Панель інструментів.

Кнопки на панелі інструментів різні для різних редакторів CoDeSys. Призначення цих кнопок приводяться в описах редакторів.

Організатор об'єктів.

Організатор об'єктів (рис. 1.2) завжди перебуває в лівій частині головного вікна CoDeSys. У нижній частині організатора об'єктів перебувають вкладки:

-  – POUs (програмні компоненти);
-  – Data types (типи даних);
-  – Visualizations (візуалізації);
-  – Resources (ресурси).

Перемикатися між відповідними об'єктами можна за допомогою мишки або клавіш переміщення.

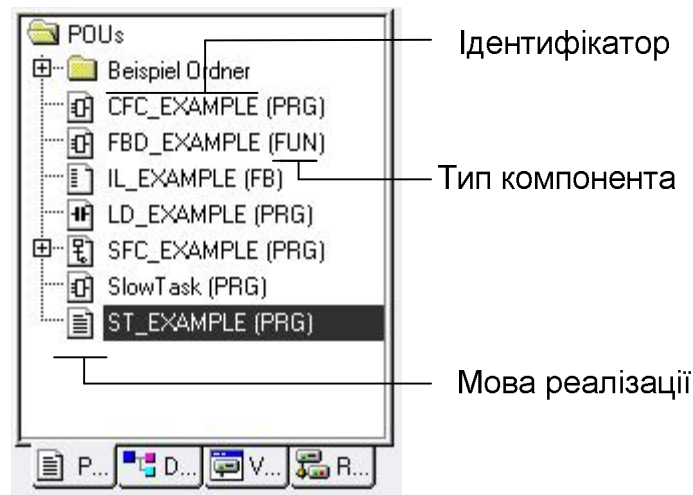


Рисунок 1.2 – Вистава програмних компонентів в організаторі об'єктів

Роздільник екрана.

Роздільник екрана – це границя між двома непересічними вікнами. В CoDeSys є наступні роздільники:

- між організатором об'єктів і робочою областю,
- між розділом оголошень і розділом коду POU,
- між робочою областю й вікном повідомлень.

Роздільники можна переміщати за допомогою мишки, натиснувши її ліву кнопку.

Робоча область.

Робоча область перебуває в правій частині головного вікна CoDeSys. Усі редактори, а також менеджер бібліотек відкриваються саме в цій області. Ім'я відкритого об'єкта перебуває в заголовку вікна.

Вікно повідомлень.

Вікно повідомлень призначене для виводу повідомлень компілятора, результатів пошуку й списку перехресних посилань.

Відкрити об'єкт, до якого ставиться дане повідомлення, можна подвійним клацанням мишкою на повідомленні.

За допомогою команд Edit ► Next error і Edit ► Previous error можна швидко переміщатися між повідомленнями про помилки.

Вікно повідомлень можна прибрати або включити за допомогою команди контекстного меню Message.

Статусний рядок.

Статусний рядок перебуває в нижній частині головного вікна CoDeSys і надає інформацію про проект і команди меню. При виборі пункту меню його опис з'являється в лівій частині рядка статусу.

За допомогою статусного рядка в режимі online можна визначити, у якому стані перебуває програма:

- SIM – у режимі емуляції.
- RUN – програма запущена.
- BP – установлена точка останову.
- FORCE – відбувається фіксація змінних.

Контекстне меню.

Замість того, щоб викликати команди з головного меню, можна скористатися контекстним меню, яке зазвичай, викликається правою кнопкою миші.

1.2 Компонентна організація проекту

Сполучення різних мов МЕК забезпечується на рівні компонентів. Саме тут утворюється код прикладної програми для ПЛК.

Компонент має властивість *інкапсуляції* – він працює як «чорний ящик», приховуючи деталі реалізації. Для роботи з компонентом досить знати його інтерфейс, що включає опис входів і виходів. Внутрішній його устрій знати не обов'язково. У графічній формі вистави компонент виглядає як прямокутник із входами ліворуч і виходами праворуч. Локальні (внутрішні) змінні компонента недоступні ззовні й у графічній виставі не відображаються.

Завдяки інкапсуляції компонента успішно вирішуються завдання *структурної декомпозиції проекту*. При цьому на верхньому рівні вистави зазвичай використовуються великі компоненти, тому що зайві подробиці на цьому рівні тільки заважають розумінню проблеми.

Ще одне завдання, що розв'язується завдяки компонентам, є *локалізація імен змінних*. Це означає, що в різних компонентах можна використовувати повторювані імена. Так, наприклад, широко розповсюджену змінну з оригінальним ідентифікатором «X» можна використовувати в кожному компоненті, і щораз це буде нова змінна. Область видимості локальних змінних визначається рамками одного компонента. Екземпляри функціональних блоків, оголошені усередині інших компонентів, також мають локальну область видимості. Однак програми й функції завжди визначені глобально.

Створення проекту починається з першого програмного компонента – POU (Program Organization Unit), який поміщається в новий проект автоматично й одержує назву PLC_PRG. Саме з нього будуть викликатися інші програмні блоки – програми, функції й функціональні блоки.

Команди керування проектом перебувають у пунктах меню File і Project. Частина команд доступна через іконки на панелі керування.

Створення й відкриття проекту.

File ► New: створює новий проект із іменем “Untitled”. При збереженні це ім'я бажано замінити.

File ► New from template: відкриває шаблон проекту. Новий проект одержує ім'я “Untitled”.

File ► Open: відкриває раніше збережений проект. Якщо в момент виклику цієї команди якийсь проект уже відкритий і в нього були внесені зміни, то CoDeSys запитає, чи потрібне зберегти цей проект чи ні.

У діалогові вікні відкриття проекту можна вибрати *проект* (файл із розширенням *.pro*) або *бібліотеку* (файл із розширенням *.lib*).

Створення нового компонента.

Створення нового програмного компонента в CoDeSys виконується командою Add Object....

Ім'я нового POU визначається в полі Name of the new POU діалогового вікна New POU (рис. 1.3). Ім'я повинне бути унікальним.

При завданні імені слід дотримуватися наступних правил:

- Ім'я не повинне містити пробілів.
- Ім'я нового *POU* не повинне збігатися з іменами інших POU.
- Ім'я нового *типу даних* не повинне збігатися з іменами інших POU або типів даних.
- Ім'я нового *списку глобальних змінних* не повинне збігатися з іменами інших списків глобальних змінних
- Ім'я нової *дії* не повинне збігатися з іменами інших дій у цьому ж POU.
- Ім'я нової *візуалізації* повинне бути оригінальним.

У всіх інших випадках ім'я буде визнано коректним.

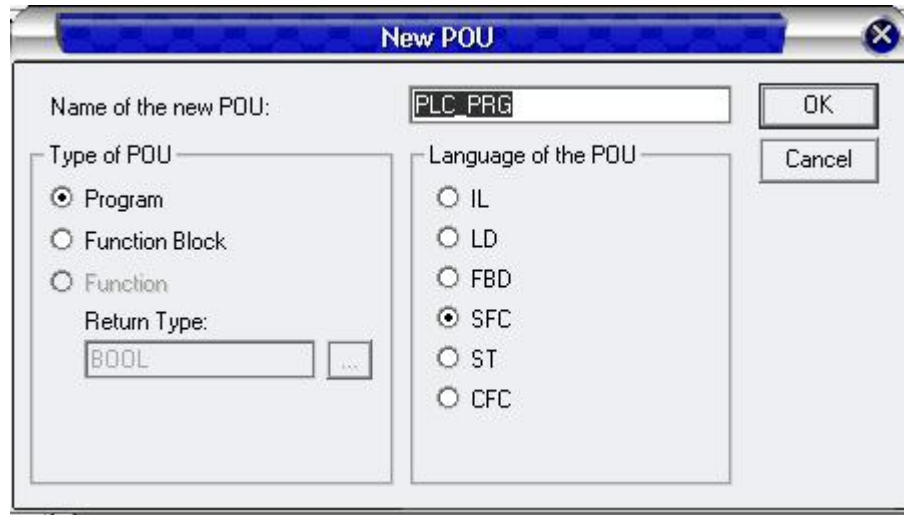


Рисунок 1.3 – Діалогове вікно New POU

При створенні POU у діалогові вікні New POU потрібно задати тип POU (програма, функція або функціональний блок) і мову, на якій цей POU буде виконаний. Тип POU задається в розділі Type of the POU, а мова – у розділі Language of the POU. Якщо POU є функцією, то також потрібно задати тип значення, що повертається, у полі Return Type. Тут можна вводити будь-який простий або обумовлений користувачем тип (масив, структура, перерахування). При цьому зручно користуватися інструментом Input assistance, який викликається клавішею <F2>.

Якщо ім'я POU задане вірно, то кнопка OK стає доступною. При натисканні OK створюється новий об'єкт в організаторі об'єктів. При цьому відкривається вікно редактора для цього об'єкта.

При вставці об'єкта з буфера за допомогою команди Edit ► Insert діалогове вікно New POU не з'являється. Якщо ім'я об'єкта, що вставляється, конфліктує з наявним об'єктом, то до нього буде доданий номер, наприклад, Righ_1.

Заповнення розділу оголошень POU.

Реалізації будь-якого POU завжди повинен передувати розділ оголошень. Оголошення функції, функціонального блоку й програми починаються, відповідно, із ключових слів FUNCTION, FUNCTION_BLOCK і PROGRAM. За ним іде ідентифікатор (ім'я компонента). Далі визначається інтерфейс POU. До інтерфейсу компонента ставляться входи VAR_INPUT, виходи VAR_OUTPUT і змінні типу вхід-вихід VAR_IN_OUT. Завершують розділ оголошень локальні змінні VAR.

У функціях розділи VAR_OUTPUT і VAR_IN_OUT відсутні. Виходом функції служить єдина змінна, що збігається з іменем функції. Тип значення, що повертається, вказується при визначенні ідентифікатора через двокрапку.

Наприклад: FUNCTION iNearby : INT

Структура розділу оголошень POU показана в таблиці 1.1.

Таблиця 1.1 – Структура розділу оголошень

Тип POU	Програма	Функціональний блок	Функція
Найменування й тип	PROGRAM ім'я	FUNCTION_BLOCK ім'я	FUNCTION ім'я: ТИП
Інтерфейс	VAR_INPUT	VAR_INPUT	VAR_INPUT
	VAR_OUTPUT	VAR_OUTPUT	
	VAR_IN_OUT	VAR_IN_OUT	
Локальні змінні	VAR	VAR	VAR

Розділ оголошень може бути обмежений тільки локальними змінними VAR.

Оголошення змінних бібліотечних POU робити не потрібно – вони втримуються в самих файлах бібліотек. Бібліотеки підключаються до проекту за допомогою менеджера бібліотек (Library Manager).

В CoDeSys є можливість автоматично конвертувати компонент із однієї мови на іншу. Автоматичний переклад можливий на мови IL, FBD і LD. Конвертування виконується командою ProjectObject ► Convert.

У графічній виставі компонента CoDeSys відбиває параметри VAR_IN_OUT зліва, постачаючи їх спеціальним значком ▷ «трикутник». У деяких системах такі параметри відображаються одночасно із двох сторін графічного блоку, тобто сполучна лінія як би проходить через зображення компонента.

Глобальні змінні VAR_GLOBAL, VAR_ACCESS і константи в CoDeSys оголошуються тільки в ресурсах. Вони існують у єдиному екземплярі й доступні на запис і читання для всіх POU проекту.

Глобальні змінні, які використовуються усередині компонента, перелічуються в його розділі оголошень під заголовком VAR_EXTERNAL. В CoDeSys це робити не обов'язково.

Інтерфейс компонента.

Інтерфейс компонента утворюється вхідними й вихідними змінними. Інтерфейсні вхідні змінні називають *формальними параметрами*. При використанні компонента його формальні параметри зв'язуються з *актуальними параметрами*. І нарешті, при виклику параметри компонента здобувають актуальні або *поточні значення*. Ці поняття необхідні для виключення двозначності при описі техніки роботи з компонентами.

Візьмемо, наприклад, стандартний блок R_TRIG. Він має вхід синхронізації з назвою CLK. При виклику блоку на вхід CLK потрібно подати

змінну, наприклад, bPulse. Далі програма компілюється й завантажується в контролер. Змінна bPulse набуває деякого значення, наприклад TRUE. Вхід CLK, природно, теж буде мати значення TRUE. Тут очевидні вже практичні відмінності. CLK – це формальний параметр, bPulse – актуальний параметр, а TRUE – це фактичне значення. З формальними параметрами доводиться мати справу при проектуванні POU і описі його інтерфейсу. Актуальні параметри працюють при використанні компонента. Поточні значення народжуються тільки в процесі виконання, тобто в «залізі».

Формальні вхідні параметри VAR_INPUT передаються POU за значенням шляхом копіювання. При виклику блоку такій змінній можна привласнити значення іншої змінної (сумісного типу), константи або вираження. Застосовуються в будь-яких POU. Можуть мати значення за замовчуванням. Відбиваються в графічній виставі з лівої сторони компонента.

Формальні вихідні параметри VAR_OUTPUT відбивають результати роботи компонента. Передаються з POU за значенням шляхом копіювання. Читання значення виходів зазвичай має сенс після виконання блоку. Поза компонентом параметри VAR_OUTPUT доступні тільки по читанню. Не використовуються у функціях, оскільки функція має тільки одне значення, що вертається. Можуть мати початкові значення. Відбиваються в графічній виставі праворуч.

Параметр типу VAR_IN_OUT одночасно є входом і виходом. Передача змінної екземпляру блоку виконується по посиланню. Це означає, що зовнішня змінна як би сама працює усередині блоку на правах внутрішньої змінної. У компонент передається тільки адреса її розташування в пам'яті даних. Для змінної VAR_IN_OUT не можна:

- використовувати у функціях;
- привласнювати початкове значення;
- звертатися як до елемента структури даних, через крапку;
- привласнювати константу.

Як і глобальні змінні, параметри VAR_IN_OUT порушують ідеологію незалежності компонентів. Правильний компонент не повинен мати можливості зіпсувати чужу пам'ять. Тому застосовувати їх потрібно дуже акуратно й тільки у випадках, коли це дійсно необхідно.

Локальні змінні VAR доступні тільки усередині компонента, поза компонентом доступу немає. Можуть мати початкові значення. Для функцій локальні змінні розміщуються в динамічній пам'яті (зазвичай в стеці). По закінченню роботи функції пам'ять звільняється й може використовуватися в інших функціях. У програмах і екземплярах функціональних блоків змінні VAR зберігають свої значення між викликами програм і екземплярів. У графічній виставі компонента локальні змінні не відбиваються.

1.3 Настроювання опцій проекту

Основні настроювання системи CoDeSys виконуються у вікні опцій, яке відкривається за допомогою команд Project ► Options. Опції діляться на кілька *категорій* (рис. 1.4). Вибір потрібної категорії проводиться в лівій частині діалогового вікна за допомогою мишки. Опції, відповідні до обраної категорії, відображаються в правій частині вікна.

Загальні настроювання зберігаються у файлі CoDeSys.ini і відновлюються при наступному запуску CoDeSys.

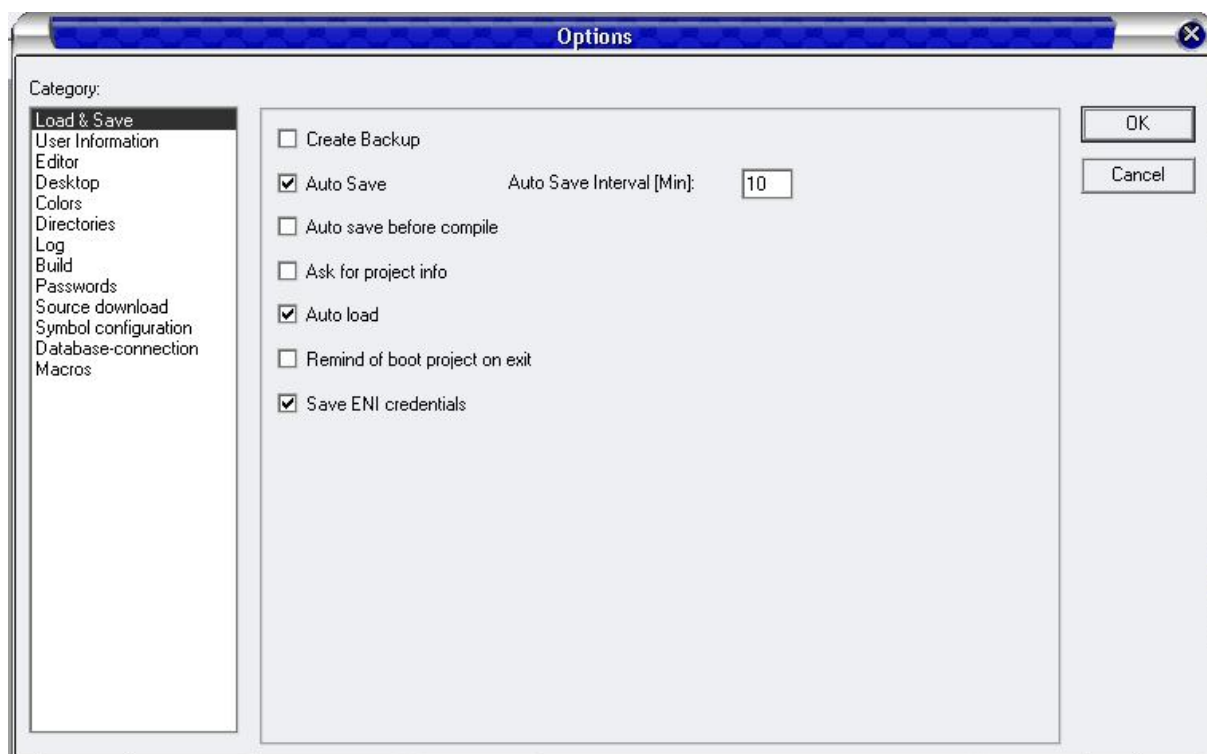


Рисунок 1.4 – Діалогове вікно Options у категорії Load&Save

Категорія Load&Save (Відкриття й збереження).

При виборі категорії Load&Save у правій частині діалогового вікна Options відображаються наступні опції:

- Create Backup указує CoDeSys створювати резервний файл із розширенням .bak при кожному збереженні. На відміну від .asd файлу (див. нижче) він не видаляється при закритті проекту.
- Autosave указує CoDeSys періодично зберігати проект у тимчасовому файлі з розширенням .asd у робочій директорії проекту.
- Auto save before compile створює аналогічні тимчасові файли перед кожною компіляцією проекту.
- Якщо обрана опція Ask for project info, то при збереженні проекту під новим іменем автоматично викликається діалогове вікно для введення інформації про проект. Це ж вікно з'являється при виборі команди Project ►

Project info.

- Опція Auto Load забезпечує автоматичне відкриття останнього відкритого проекту при запуску CoDeSys.

- Remind of boot project on exit: Якщо проект був змінений і завантажений у контролер без завантажувального проекту, то при спробі закрити проект ви одержите нагадування: "Після останнього завантаження завантажувальний проект не був створений. Закрити проект? " (No boot project created since last download. Exit anyway?).

- Save ENI credentials (зберігати посвідчення користувача). Login бази даних ENI буде зберігатися в проекті.

Категорія User information (Інформація про користувача).

При виборі категорії User information у правій частині діалогового вікна Options можна ввести своє ім'я (Name) і ініціали (Initials), а також назву компанії (Company). Кожне з полів вільно редагується. Задана інформація буде автоматично вставлена в усі нові проекти, створені на даному комп'ютері.

Категорія Editor (Редактор).

У категорії Editor (рис. 1.5) можна встановити:

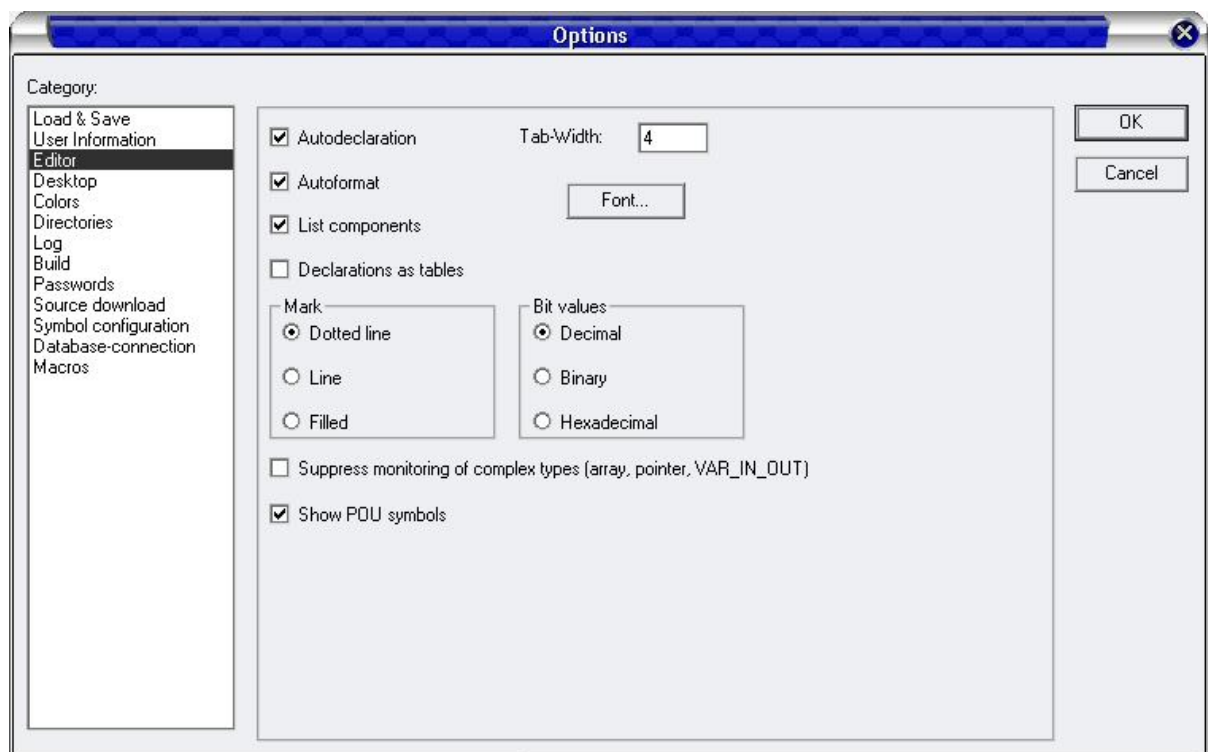


Рисунок 1.5 – Діалогове вікно Options у категорії Editor

- Autodeclaration: при введенні імені нової змінної автоматично буде запропонований діалог для її оголошення.

- Autoformat: CoDeSys буде автоматично виконувати форматування тексту в ІЛ редакторі й розділах оголошень.

- List components: включає функцію інтелектуального аналізу введення (Intellisense). Працює це так: у позиції, куди необхідно вставити ідентифікатор, ставиться крапка. Потім відкривається список глобальних змінних проекту. Після введення імені екземпляра функціонального блоку, буде відкритий список усіх входів і виходів блоку. Функція Intellisense доступна в редакторах, у менеджеріві рецептів, у візуалізації й трасуванню.

- Declarations as tables: розділ оголошень буде відображатися у вигляді карток з таблицями.

- Опції Mark визначають, як буде виглядати виділення в графічних редакторах. Якщо обрана опція Dotted, то курсор – це прямокутник з пунктирною границею, якщо Line, то курсор – це прямокутник із суцільною границею, якщо Filled – прямокутник, зафарбований чорним.

- Опції Bitvalues визначають систему числення (за замовчуванням) для відображення значень типу змінних – бітових рядків (BYTE, WORD і DWORD), двійкових чисел (Binary), шістнадцятерічних (Hexadecimal) і десяткових (Decimal).

- Suppress monitoring of complex types (Array, Pointer, VAR_IN_OUT): Якщо дана опція активна, то складні дані, такі як масиви, покажчики й VAR_IN_OUT не будуть відображатися у вікні online моніторингу.

- Show POU symbols: Якщо дана опція активна, то в рамці програмного компонента в графічних редакторах буде відображатися відповідна іконка. Для цього її зображення у вигляді однойменного bmp-файлу повинне бути присутнім у директорії бібліотеки.

Desktop (Робітничий стіл).

У діалоговому вікні Desktop проводяться наступні налаштування:

- Tool bar і Status bar указують, чи відображати панель інструментів (під головним меню) і статусний рядок.
- Show print area margins: у вікнах редактора будуть показані обмежники для параметрів сторінки при печатці.
- MDI representation указує, чи використовувати MDI інтерфейс вікна CoDeSys, опція активна за замовчуванням. При відключеній опції використовується SDI інтерфейс.

Colors (Кольори).

Діалогове вікно Colors показано на рисунку 1.6.

У категорії Colors можна редагувати кольорні установки CoDeSys. Тут можна змінити колір для номерів рядків (Line numbers), для поточних позицій (Current position), для точок останова (Breakpoint positions), для установлених точок останова (Set breakpoint), для пройдених позицій (Reached Positions), а також колір при моніторингу значень логічних змінних (Monitoring of Bool).

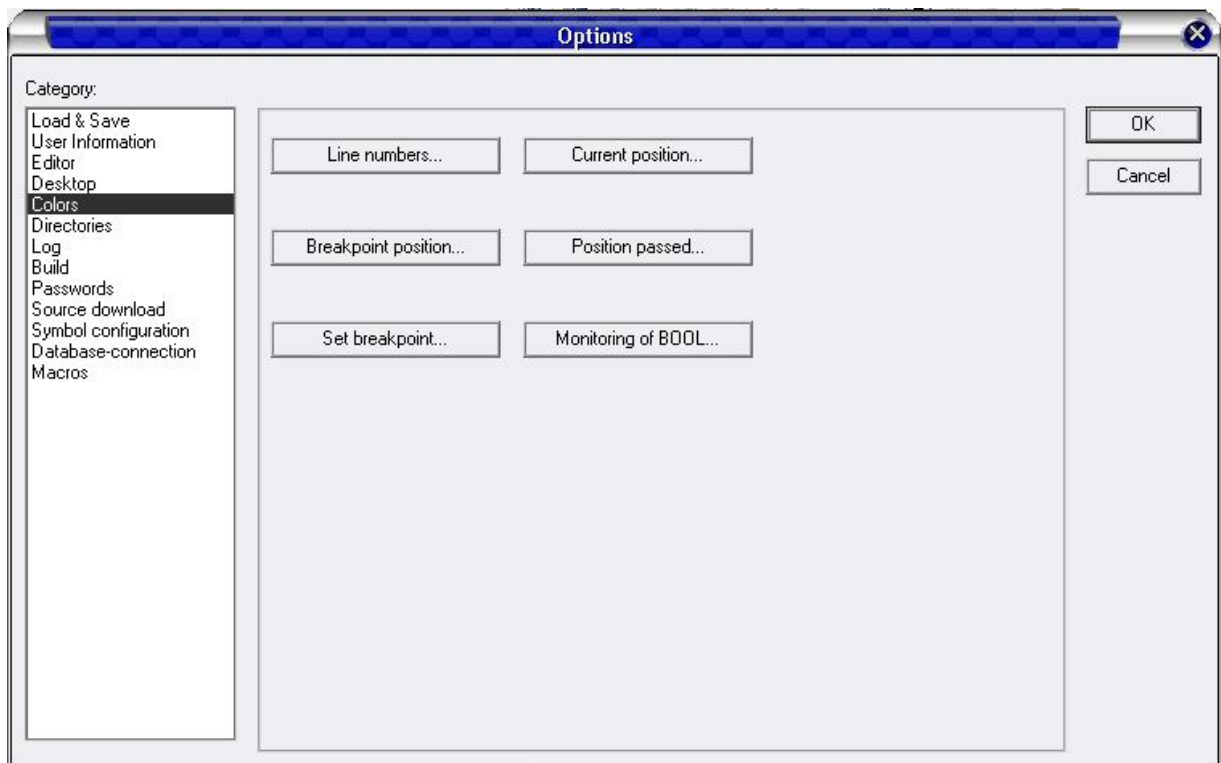


Рисунок 1.6 – Діалогове вікно категорії Colors

За замовчуванням установлені наступні кольори:

- номер рядка – світло-сірий;
- поточна позиція – червоний;
- точки останова – темно-сірий;
- установлена точка останова – блакитний;
- пройдена позиція – зелений;
- колір при моніторингу значень логічних змінних – синій.

Вибір кольору здійснюється у стандартному діалоговому вікні.

Directories (Директорії).

У діалогові вікні Directories слід увести директорії, у яких перебувають бібліотеки (Libraries), файли конфігурації контролерів (Configurations files) і файли візуалізації (Visualization files). Також потрібно вказати директорії, у яких будуть зберігатися файли компілятора (Compile files), наприклад, map- і list-файли, а також файли, завантажені з контролера (Upload files).

Log (Бортжурнал).

У цьому діалозі можна настроїти бортжурнал (*.log-файл), у який записуються всі дії користувача й дії, виконувані CoDeSys під час режиму Online.

Build (Генератор коду).

Діалогове вікно Build показано на рисунку 1.7.

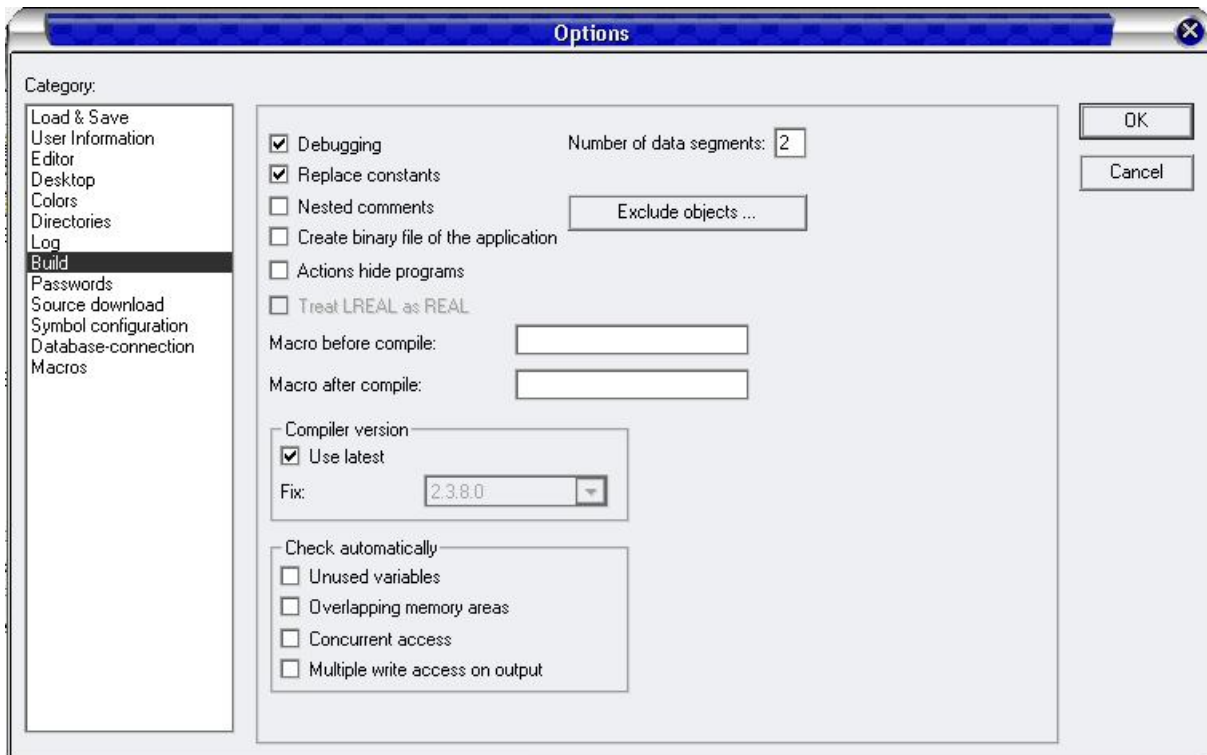


Рисунок 1.7 – Діалогове вікно Build

Активність категорії Debugging залежить від установленної цільової платформи. Вона включає генерацію додаткового коду для виконання розширених функцій налагодження (наприклад, установка точок останова).

У правій частині вікна є наступні поля:

- **Replace constant:** дозволяє «вшивати» значення констант у машинний код.
- **Nested comments:** дозволяє використовувати вкладені коментарі.
- **Create binary file of application:** при компіляції буде створений файл, що містить двійковий код додатка. Такий файл має ім'я <ім'я_проекту>.bin.
- **Actions hide programs:** активується за замовчуванням при створенні нового проекту. Опція означає, що у випадку збігу імені локальної дії з іменем глобальної змінної або програми, установлюється наступна ієрархія доступу – локальна змінна, локальна дія, потім глобальна змінна й програма.
- **Treat LREAL as REAL:** змушує компілятор використовувати тип REAL для LREAL оголошень.
- **Число Number of Data** визначає, скільки сегментів пам'яті розміщується в контролері під дані.
- **Клавіша Exclude objects** відкриває діалог виключення об'єктів з компіляції (Exclude objects from build). Компоненти (POU), які не потрібно компілювати, виключаються установкою опції Exclude. Виключені POU будуть відображатися зеленим кольором.

- **Compiler Version:** Тут можна вибрати версію компілятора. За замовчуванням, установлений прапорець Use latest, що означає використання новітньої версії компілятора.

Усі опції, установлені в цій категорії, зберігаються в проекті.

Passwords (Паролі).

Для захисту файлів від несанкціонованого доступу можна встановити паролі.

Source download (Завантаження вихідних файлів).

У цій категорії група опцій Extent дозволяє вказати, які вихідні файли повинні завантажуватися в контролер.

Опція Sourcecode only включає в завантаження тільки файл проекту. Опція All files, крім того, включає необхідні бібліотеки, файли конфігурації, візуалізації і т.д.

Група опцій Timing управляє порядком завантаження. Опція Implicit at load задає безумовне завантаження вихідних файлів по команді Online ► Download. Опція Notice at load приводить до виникнення запиту про необхідність завантаження вихідних файлів при завантаженні коду. Опція On demand дозволяє завантажувати вихідні файли тільки по команді Online ► Sourcecode download.

Symbol Configuration (Символьна конфігурація).

У цій категорії можна настроїти символний файл (текстовий файл із розширенням *.sym і двійковий з розширенням *.sdb). Це необхідно для обміну даними з контролером через символний інтерфейс.

Project source control (Контроль текстів проекту).

Цей діалог включає підтримку керування проектом через ENI інтерфейс і визначає зв'язок з базою даних.

Macros (Макроси).

У цьому діалоговому вікні можна створити макроси (макрокоманди), які складаються з команд пакетного механізму CoDeSys. Створені макроси будуть додані як команди в меню Edit ► Macros.

1.4 Команди керування проектом і об'єктами

Керування об'єктами проекту.

Далі будуть наведені відомості про принципи роботи з *об'єктами*, що визначають структуру проекту.

Слово «об'єкт» використовується як загальне поняття, що включає програмні компоненти (POU), типи даних, візуалізації (visualizations), розділи глобальних (global) і конфігураційних змінних (variable configuration), трасування (Sampling trace), конфігурацію контролера (PLC configuration), конфігурацію завдань (Task Configuration) і менеджер рецептів (Watch and

Receipt Manager). Для структурування проекту використовуються папки. Усі об'єкти проекту відбиваються в *організаторові* об'єктів.

Об'єкти в організаторові об'єктів можна перетягувати мишкою (drag&drop). Якщо виникає конфлікт імен, то до імені нового об'єкта буде доданий номер, наприклад, "Object_1".

Робота з об'єктами здійснюється вибором команд Project ► Object. У списку, що відкривається, є наступні функції.

Delete. Швидкий виклик: <Delete>.

Видаляє обраний об'єкт або папку з усіма об'єктами, що входять в папку. Якщо при видаленні об'єкт був відкритий у редакторі, то цей редактор буде автоматично закритий.

Add. Швидкий виклик:<Insert>.

Створює новий об'єкт. Тип створюваного об'єкта залежить від обраної в організаторові об'єктів вкладки. Для об'єктів типу Global Variables, Data types, Function, Function Block або Program доступне використання шаблонів.

Rename. Швидкий виклик: <Пробіл>.

Перейменовує обраний об'єкт або папку. Нове ім'я повинне бути унікальним. Якщо при перейменуванні було відкрите вікно редагування об'єкта, то заголовок вікна зміниться автоматично.

Convert. Ця команда діє тільки на ROU. Вона конвертує ROU з будь-якої мови на один із трьох мов IL, FBD і LD. Перед використанням команди потрібно скопіювати проект. Для цього у вікні Convert Object потрібно вибрати один із трьох зазначених вище мов і дати ROU нове ім'я. Новий ROU буде доданий у список ROU.

Copy. При виконанні цієї команди об'єкт копіюється й зберігається під новим іменем. Ім'я нового ROU уводиться в діалогові вікні Copy Object. Аналогічну функцію виконує команда Edit ► Copy, але діалогове вікно при цьому не з'являється.

Properties. Команда відкриває діалог властивостей об'єкта Properties, обраного в організаторові об'єктів. Вид діалогу визначається типом об'єкта:

- для списків глобальних змінних буде відкритий діалог Global variable list. У ньому задаються параметри відновлення списку й, якщо можливо, параметри обміну даними за допомогою мережних змінних.
- на вкладці Visualization визначається спосіб використання об'єкта візуалізації.

При роботі з папками використовуються наступні команди.

New Folder. Команда додає нову папку в організаторові об'єктів. Якщо при виклику цієї команди була виділена папка, то нова буде створена в ній. А якщо ні, то вона створюється на тому ж рівні вкладеності. Якщо виділена дія, то папка створюється на рівні вкладеності, до якого належать дії. Ця команда доступна, коли обрано об'єкт в організаторові об'єктів. Вона втримується в

контекстному меню, яке викликається натисканням клавіш <Shift>+<F10> або правою кнопкою миші.

Знову створена папка одержує ім'я New Folder. Папки можна називати за наступними правилами:

- Папки на одному рівні вкладеності повинні мати різні імена. На різних рівнях допускаються однакові імена.
- Папка не повинна мати таке ж ім'я, як і який-небудь об'єкт на цьому рівні.

Якщо при виконанні команди папка з іменем New Folder уже існує, то нова папка одержить таке ж ім'я тільки з номером, наприклад, New Folder_1.

Expand nodes. За допомогою цієї команди папку можна відкрити.

Collapse nodes. Команда закриває папку.

Ці команди доступні, коли обрано об'єкт в організаторові об'єктів. Команди перебувають у контекстному меню.

1.5 Приклад створення проекту в CoDeSys

Для освоєння методики створення проекту в CoDeSys використаємо простий приклад, наведений Петровим І.В. (<http://www.prolog.smolensk.ru>).

Приклад завдання проекту.

Нехай робочий орган деякого механізму робить циклічний рух по периметру прямокутника. Оператор повинен періодично підтверджувати правильність функціонування механізму. Якщо оператор не дає підтвердження, система видає попередження, а потім припиняє роботу механізму. Таким чином, завдання проекту включає дві підзадачі – контроль коректності роботи механізму (підтвердження оператора) і керування циклом роботи механізму.

Запуск CoDeSys.

CoDeSys запускається точно так, як більшість Windows додатків: Пуск ► Програми ► 3S Software ► CoDeSys V2.3.

Створення проекту для оцінки коректності роботи механізму.

Новий проект створюється командою File ► New. При цьому виводиться діалогове вікно *Target Settings* (рис. 1.8), у якому вибирається цільова платформа майбутньої програми.

Проект є машинно-незалежним, тобто його можна випробувати в режимі емуляції (варіант None). Але для визначеності виберемо конкретний контролер. На сторінці діалогового вікна Configuration установимо: 3S CoDeSys SP RTE і підтвердимо введення – ОК.

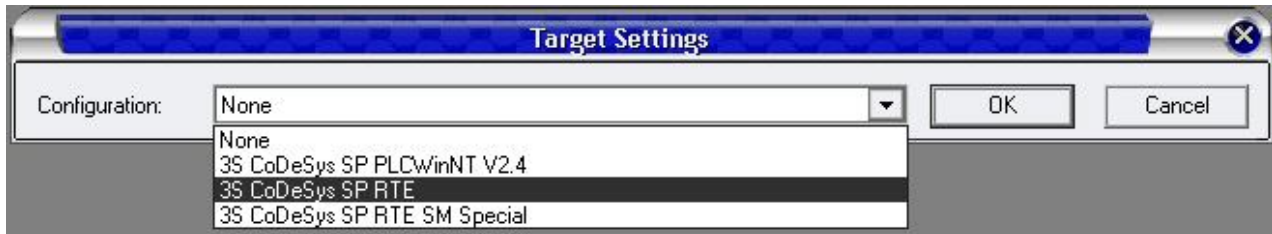


Рисунок 1.8 – Діалогове вікно для настроювання цільової платформи

Головна програма PLC_PRG POU.

Наступне діалогове вікно (New POU) визначає тип першого програмного компонента. Слід урахувати, що оператор повинен підтверджувати роботу саме перемиканням клавіші, а не просто спати з постійно натиснутою клавішею підтвердження. Для визначення моментів натискання й відпускання, тобто моментів переходів значення логічної змінної з нуля (FALSE) в одиницю (TRUE) і навпаки доцільно використовувати готові функціональні блоки мови FBD.

На сторінці Language of the POU виберемо мову реалізації FBD і збережемо запропоновані за замовчуванням тип компонента (Type of the POU Program) – програма й ім'я (Name) – PLC_PRG.

PLC_PRG – це особливий програмний компонент (POU). В однозадачних проектах він *циклічно* викликається системою виконання.

Оголошення перемикача підтвердження.

Перемикач підтвердження – це змінна, яка при підтвердженні оператором коректності роботи механізму буде змінювати своє значення. Для оголошення цієї змінної виділимо рядок питань ??? у першому ланцюзі графічного FBD редактора й уведемо найменування змінної. Нехай це буде Observer (спостерігач). Далі потрібно натиснути на клавіатурі стрілку вправо й у діалогові вікні визначення змінної, що відкрилося, зберегти **найменування** (Name – **Observer**) і логічний **тип** (Type – **BOOL**). Клас змінної (Class) слід замінити на глобальний (VAR_GLOBAL). Усі оголошення підтвердимо натисканням ОК.

Тепер визначення змінної Observer повинне з'явитися у вікні глобальних змінних проекту (Global Variables):

```
VAR_GLOBAL
  Observer: BOOL
END_VAR
```

Детектор переднього фронту.

Повернемося у вікно редактора PLC_PRG (клацнем по кнопці POU в організаторові об'єктів або зробимо подвійний клацання по імені POU). Виділимо позицію праворуч від змінної Observer. Клацнувши далі по пунктирному прямокутнику правою кнопкою миші, у контекстному меню виберемо команду Vox.

За замовчуванням вставляється елемент AND. Для формування логічної одиниці по передньому фронту вхідного сигналу необхідний тригер. Використовуючи асистента уведення (клавішу F2), у діалогові вікні ліворуч виберемо категорію стандартних функціональних блоків – Standard Function Blocks. Зі стандартної бібліотеки у вікні праворуч (standard.lib) у групі тригерів (trigger) вибираємо R_TRIG.

Для нового екземпляра функціонального блоку R_TRIG необхідно задати ім'я. Клацнувши мишкою над зображенням тригера, уводимо ім'я – Trig1. Після цього автоматично виводиться діалогове вікно визначення змінних Declare Variable, у якому потрібно вказати клас – локальна змінна (Class – VAR), ім'я (Name – Trig1) і тип (Type – R_TRIG). Після цього натискаємо OK.

У результаті цих дій вікно редактора приймає вид, показаний на рис. 1.9.

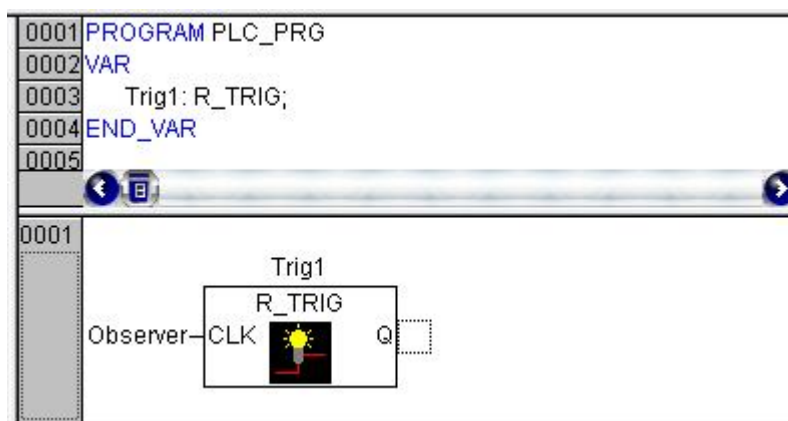


Рисунок 1.9 – Вікно редагування з детектором переднього фронту

Детектор заднього фронту.

Виділимо вихід функціонального блоку Trig1 і вставимо (як було описано вище) елемент AND. Далі перейменуємо його в OR (логічне АБО). Виділимо вільний вхід функціонального блоку OR і вставимо перед ним екземпляр функціонального блоку F_TRIG під іменем Trig2. На вхід F_TRIG, за допомогою асистента введення (F2) подамо змінну Observer (категорія Global Variables).

Детектори переднього й заднього фронтів показано на рисунку 1.10.

Контроль часу очікування підтвердження оператора.

Для контролю часу необхідний таймер. Вставимо після OR таймер із затримкою вимикання (екземпляр функціонального блоку TOF). Привласнимо йому ім'я Timer1. Три знаки питання на вході PT замінимо константою T#10s, що відповідає 10 секундам. Цей час можна міняти в процесі налагодження.

Вихідний сигнал “Попередження”.

Виділимо вихід Q таймера Timer1 і в контекстному меню (права кнопка миші) задамо команду Assign (привласнити). Замінимо питання на ім'я змінної – Warning. У діалозі визначення змінної задамо клас (Class – VAR_GLOBAL) і тип (BOOL).

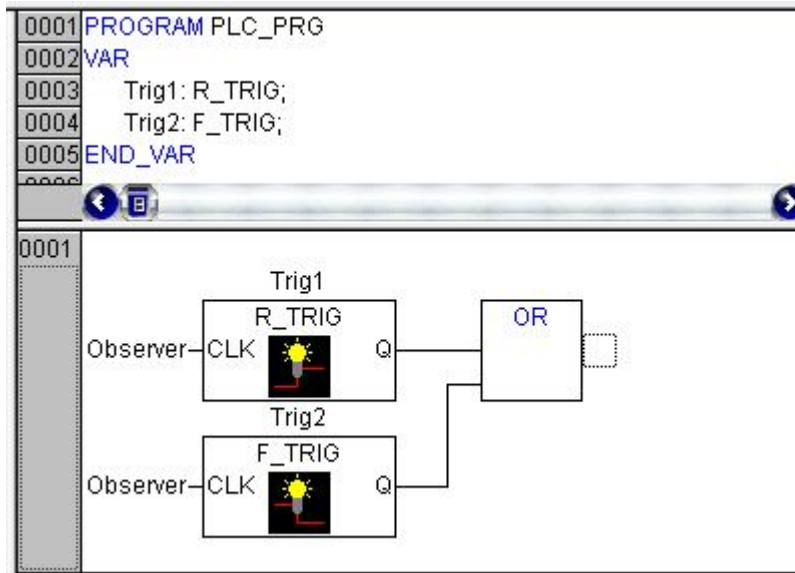


Рисунок 1.10 – Вікно редагування з детекторами переднього й заднього фронтів сигналу Observer

Тепер потрібно виділити позицію в середині лінії, що з'єднує вихід таймера й змінну Warning. У контекстному меню задамо команду Negate. Кругок, що з'явився при цьому, означає інверсію значення логічного сигналу.

Результати програмування сигналу “Warning” наведено на рис. 1.11.

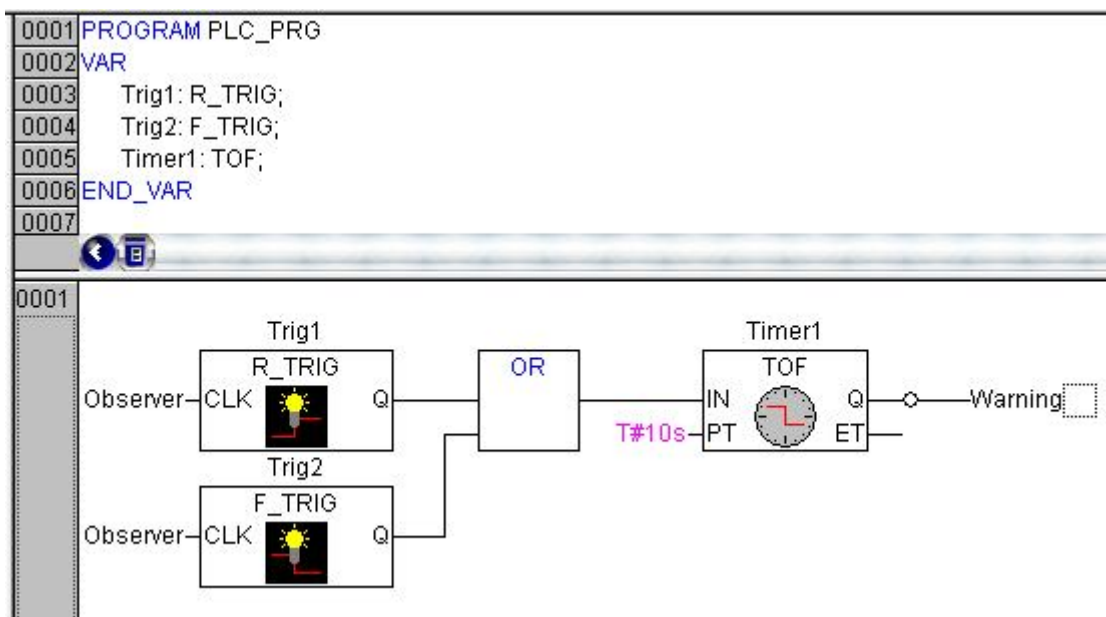


Рисунок 1.11 – Програма формування сигналу “Warning”

Формування сигналу Стоп.

Якщо після закінчення часу, що задається таймером Timer1, оператор не подав сигнал підтвердження правильності роботи механізму, контролер повинен подати команду Stop.

Створимо новий ланцюг командою меню Insert ► Network (after). У новий ланцюг вставимо елемент (Box) і виберемо зі стандартної бібліотеки тип TON (таймер із затримкою включення). Задамо йому ім'я – Timer2.

Для запуску таймера подамо на вхід IN з використанням асистента введення (F2) змінну Warning, а на вхід PT – константу T#5s. Виходу екземпляра функціонального блоку Timer2 (командою Assign) привласнимо ім'я Stop і задамо клас глобальної логічної змінної (Class – VAR_GLOBAL).

Завершальний вид програми для контролю коректності роботи механізму наведено на рисунку 1.12.

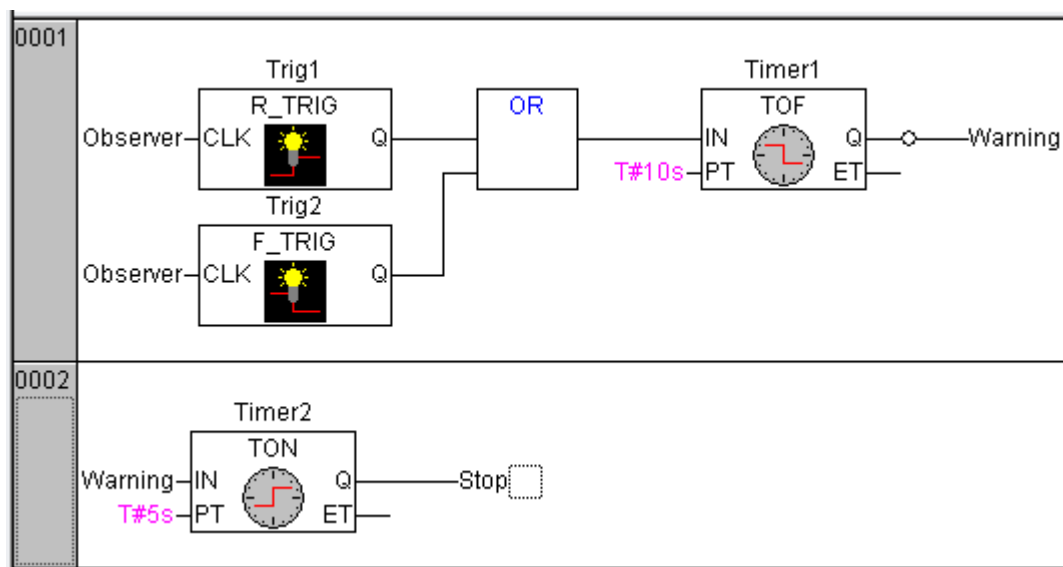


Рисунок 1.12 – Остаточний вид програми контролю коректності роботи механізму

Вставка POU керування механізмом.

У лівій частині вікна CoDeSys розташований організатор об'єктів POUs. У ньому є програма PLC_PRG. Вставимо командою Add object у контекстному меню новий програмний компонент із іменем Machine (Type of POU – program) і визначимо для нього мову SFC (Language – SFC), яка зручна для програмування циклічних процесів.

За замовчуванням, створюється порожня діаграма, що містить початковий крок Init і початковий перехід Trans0, що закінчується поверненням до Init (рис. 1.13).

Якщо праворуч від Init буде прямокутник з дією (Action), потрібно зняти в контекстному меню прапор Use Iec-steps і перевизначити заново POU Machine.

Визначаємо послідовність роботи механізму.

Кожній фазі роботи механізму повинен відповідати певний етап (крок). Якщо клацнути лівою кнопкою миші на горизонтальній рисці переходу Trans0, то перехід буде оточений пунктирною рамкою. У контекстному меню виберемо команду вставки кроку й переходу Step-Transition (after). Аналогічно

повторимо вставку ще 4 рази. Усього повинно вийти шість кроків з переходами.

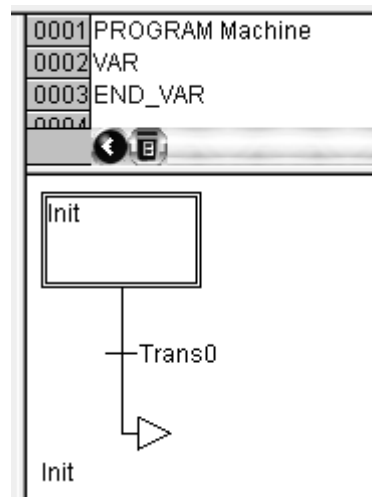


Рисунок 1.13- Початковий крок нового POU

Виділяючи клацанням миші імена кроків, визначимо нові найменування. Перший після Init крок повинен назватися Go_Right (вправо). Під ним – Go_Down (униз), Go_Left (уліво), Go_Up (нагору) і Count (рахунок).

Перехід повинен містити умову, що дозволяє перемикання на наступний крок. Перехід після кроку Init назвемо Start. Визначимо клас (Class – VAR_GLOBAL) і тип (Type – BOOL) нової логічної змінної. При одиничнім значенні цієї змінної починається цикл роботи механізму. Для наступного переходу задамо умову $X_pos = 100$. При досягненні цієї умови ввімкнеться наступна фаза руху. У якості умови третього кроку приймемо $Y_pos = 50$, четвертого – $X_pos = 0$, п'ятого – $Y_pos = 0$ і шостого – TRUE (перехід дозволений відразу ж, після однократного виконання).

Послідовність переходів і кроків мовою SFC наведено на рис. 1.14.

Програмуємо перший крок.

Клацнувши двічі на кроці Go_Right, викликаємо діалогове вікно вибору мови його реалізації (Language). Вибираємо ST (Structured Text) і переходимо в автоматично відкрите вікно текстового редактора. У цьому кроці робочий орган механізму повинен переміщатися по осі X вправо. Програма повинна виглядати так:

```
X_pos:= X_pos + 1;
```

Завершуємо введення клавішею Return і у вікні, що відкрилося, визначаємо тип змінної X_pos: INT (ціле). Після програмування кроку куточок прямокутника (кроку) повинен бути зафарбований. Це ознака того, що дія кроку визначена.

Програмуємо наступні кроки.

Повторюємо описану послідовність для всіх кроків, що залишилися. Змінні Y_pos і Counter також повинні бути типу INT.

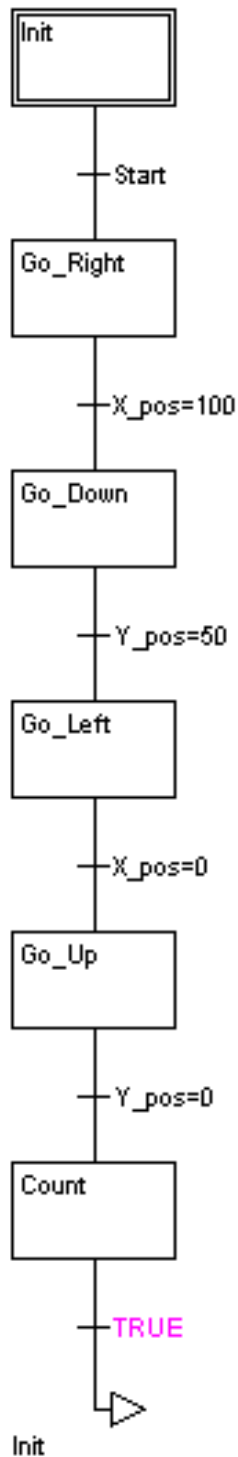


Рисунок 1.14 – Послідовність переходів і кроків мовою SFC

У наступних кроках представимо програми в такий спосіб:

Крок Go_Down: $Y_pos := Y_pos + 1;$

Крок Go_Left: $X_pos := X_pos - 1;$

Крок Go_Up: $Y_pos := Y_pos - 1;$

Крок Count: $Counter := Counter + 1;$

Останов механізму. Повернемося в організатор об'єктів до PLC_PRG ROU і додамо третій ланцюг. Замість питань вставимо змінну Stop. Потім з

контекстного меню вставимо в цей ланцюг оператор Return, який буде переривати роботу програми PLC_PRG POU при одиничнім значенні Stop. Вид цього ланцюга наведено на рисунку 1.15.

Виклик POU керування механізмом.

Для зв'язку двох створених POU додамо ще один ланцюг. Виділимо його і вставимо елемент Vox з контекстного меню. Зазвичай це буде AND. За допомогою асистента введення (натиснути F2) задамо в лівій частині діалогового вікна категорію користувацьких програм (User defined Programs), після чого в правій частині вікна виберемо POU керування механізмом – Machine.

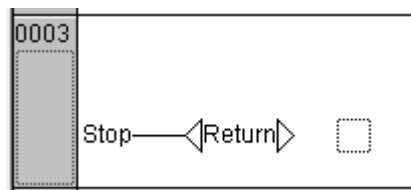


Рисунок 1.15 – Вид третьому ланцюга програми

Остаточний вид програми PLC_PRG показано на рисунку 1.16.

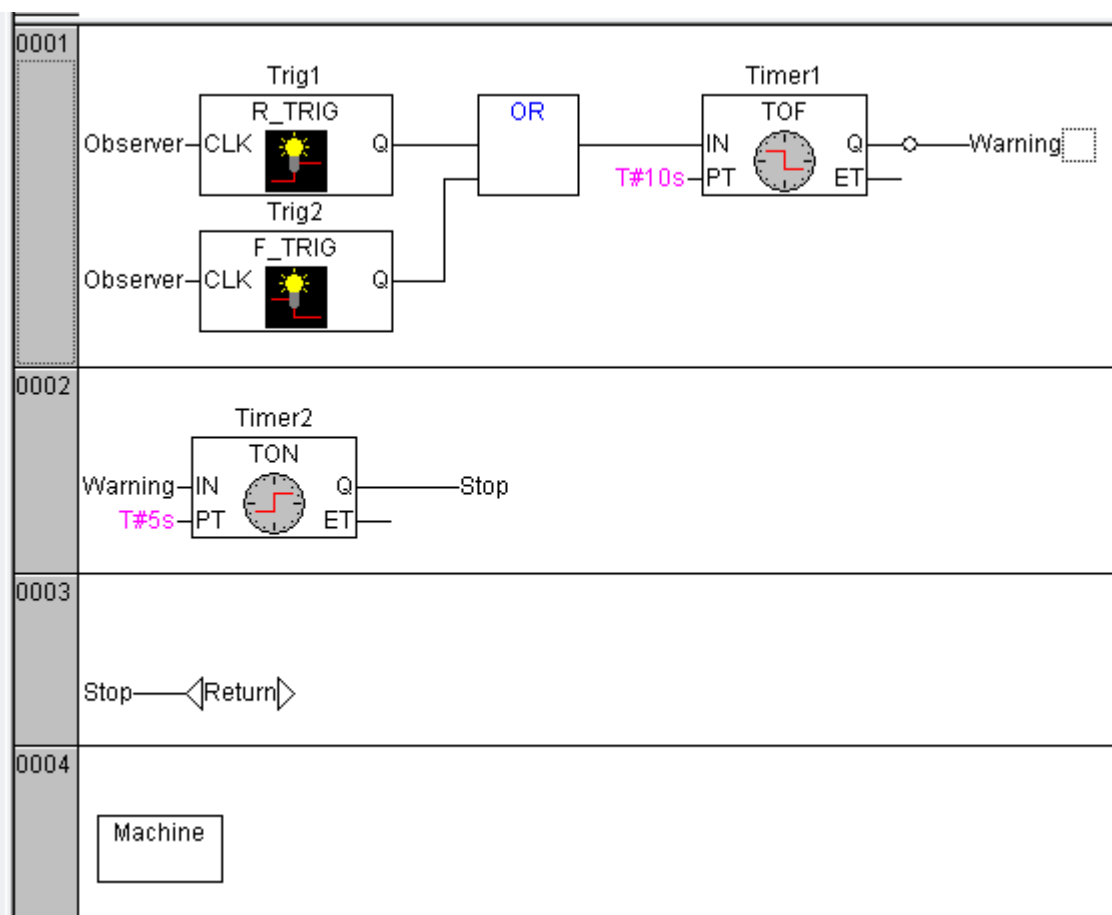


Рисунок 1.16 – Остаточний вид програми PLC_PRG

Компіляція проекту.

Компіляцію всього проекту можна виконати або командою меню Project ► Rebuild all, або клавішею F11. Якщо все зроблене правильно, то в нижній частині вікна повинне з'явитися повідомлення: 0 errors. А якщо ні, то необхідно виправити допущені помилки, розгорнуті повідомлення про яких наведені в цьому ж вікні.

1.6 Методика візуалізації проекту

Створюємо об'єкт візуалізації.

В організаторові об'єктів CoDeSys третя сторінка називається візуалізація (Visualization). Після переходу на сторінку візуалізації в контекстному меню вводимо команду додавання об'єкта Add object.

Новому об'єкту привласнимо ім'я Observation. Після натискання ОК відкривається чистий аркуш вікна редактора візуалізації.

Зображуєм перший елемент візуалізації.

Почнемо із кнопки підтвердження. Зобразимо її у вигляді прямокутника з текстом ОК. На панелі інструментів або в розділі меню Insert виберемо прямокутник (Rectangle). У вікні редактора візуалізації при натиснутій лівій кнопці миші розтягнемо прямокутник до потрібних розмірів.

Настроювання першого елемента візуалізації.

Діалогове вікно настроювання візуалізації (рис. 1.17) викликається подвійним клацанням миші на його зображенні.

Задамо в розділі Category (категорія) текст (Text), а у віконці Contents (уміст) уведемо слово “ОК”.

Далі виберемо категорію Variables (змінна). Клацнувши мишею в поле зміни кольору Change Color і скориставшись асистентом уведення (F2), вставимо змінну .Observer зі списку глобальних змінних.

Після цього переходимо в категорію Colors (колір). Задамо колір зафарбування елемента (Inside), наприклад, світло-блакитний. Для «збудженого» стану необхідно визначити інший колір (Alarm color), наприклад, блакитний. У категорії введення (Input) необхідно ще раз увести змінну Observer і поставити прапорець Toggle variable, що дозволить змінювати значення змінної кличем. Тепер діалог настроювання можна закрити.

У підсумку прямокутник буде відображатися світло-блакитним при значенні змінної Observer = FALSE і блакитним при значенні Observer = TRUE. Колір буде змінюватися при кожному “натисканні” кнопки.

Набудовуємо інші елементи візуалізації.

Для візуалізації процесу контролю створимо два індикатори – “Увага” і “Стоп”.

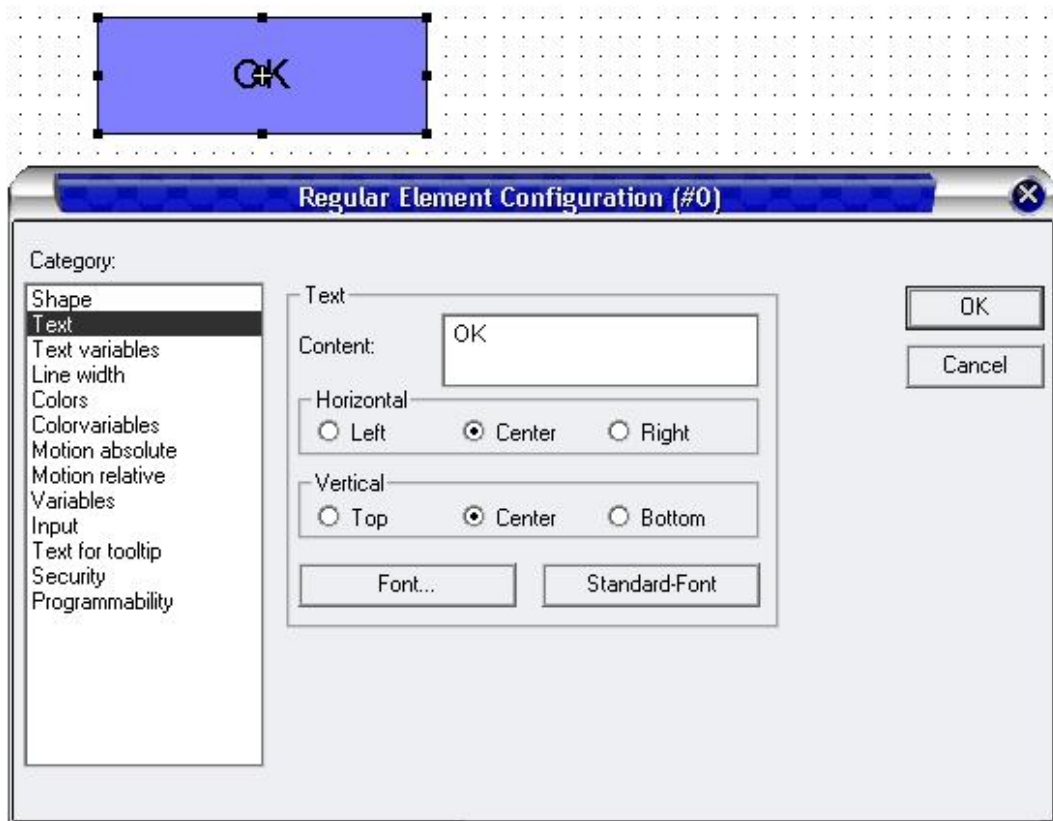


Рисунок 1.17 – Діалогове вікно настроювання візуалізації кнопки ОК

Намалюємо одну окружність (Увага), далі скопіюємо її командою Edit ► Copy й вставимо копію (Edit ► Paste) для сигналу Стоп.

У діалогові вікні настроювання в розділі Category виберемо Text, і в поле Contents уведемо текст “Увага”. У категорії Colors у розділі Color для Inside виберемо сірий колір, а в розділі Alarm color для Inside – червоний.

Виконаємо настроювання окружності Стоп:

- У категорії Text у поле Contents уведемо текст – “Стоп”.
- У категорії Variable у поле Color change вставимо за допомогою асистента введення (F2) змінну .Stop.

Для візуалізації роботи механізму виділимо окреме поле Механізм, на яким розмістимо кнопку Пуск, лічильник робочих циклів механізму (Лічильник) і деякий елемент для індикації положення робочого органа, наприклад, прямокутник.

Намалюємо прямокутник для клавіші Пуск, що має наступні настроювання:

- У категорії Text у поле Contents – текст “Пуск”.
- У категорії Variable у поле Color change – змінна .Start.
- У категорії Input установлюємо прапорець Toggle variable і вставляємо змінну .Start.
- У категорії Colors у розділі Color уводимо зафарбування елемента

(Inside) червоним, а в розділі Alarm color зеленим.

Далі малюємо прямокутник для лічильника Counter з наступними налаштуваннями:

- У категорії Text у поле Contents вводимо текст – Лічильник %s. Тут рядок %s служить для розміщення інформації про значення змінної.
- У категорії Variable у поле Textdisplay вводимо змінну Machine.Counter.

Намалюємо невеликий прямокутник, що позначає робочий інструмент механізму, з наступними налаштуваннями:

- У категорії Motion Absolute у поле X-Offset вводимо змінну Machine.X_pos, а поле Y-Offset – змінну Machine.Y_pos.
- У категорії Colors у поле Color установимо зафарбування Inside блакитним кольором.

На закінчення намалюємо дві декоративні рамки для поділу областей контролю й механізму. Задамо в них відповідні написи з вирівнюванням по низу (Vertical alignment bottom). Використовуючи контекстне меню, помістимо декоративні прямокутники на задній план (Send to back). Наприкінці роботи, вікно візуалізації буде виглядати так, як показано на рисунку 1.18.

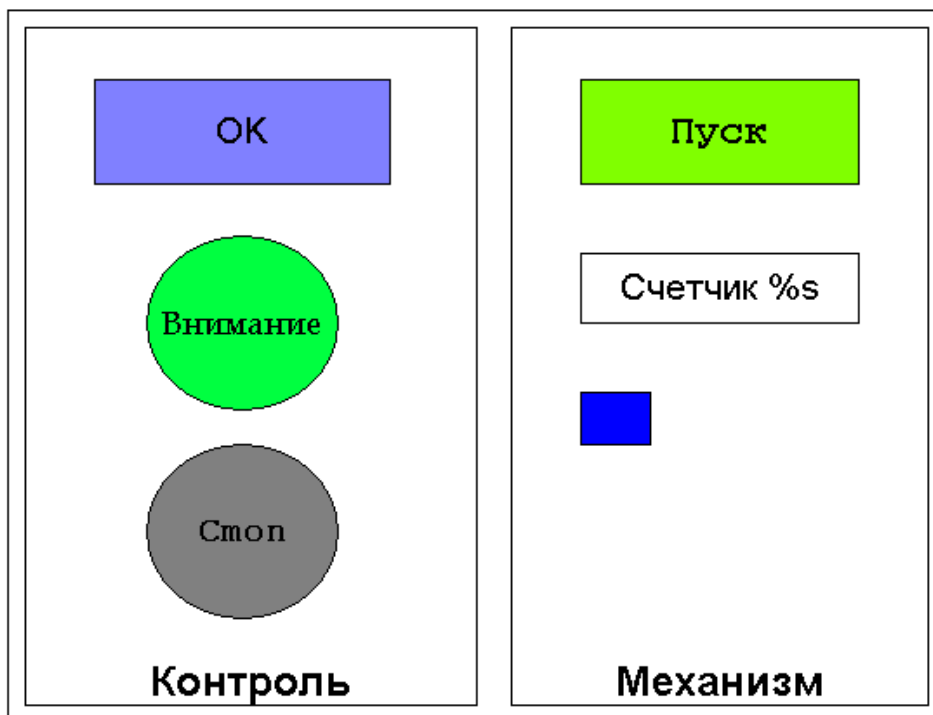


Рисунок 1.18 – Остаточний вид проекту візуалізації

Запуск проекту.

Для запуску проекту в режимі емуляції слід установити прапорець у меню Online ► Simulation Mode.

З'єднання з контролером установлюється командою Online ► Login із середовища програмування CoDeSys. Якщо використовується вилучене з'єднання, CoDeSys попросить підтвердити завантаження (download) коду проекту.

Команда Online ► Run запускає проект. Перейшовши у вікно візуалізації, можна перевірити роботу механізму.

1.7 Варіанти завдань і зміст звіту по роботі

Варіанти завдань до роботи наведені в Додатку А.

Звіт по роботі повинен містити завдання, програмні компоненти, ілюстрації процесу програмування. При захисті звіту необхідно продемонструвати роботу програми за допомогою розроблених засобів візуалізації й відповісти на запитання, що стосуються інтерфейсу програмної системи CoDeSys і приймань програмування.

2 РОЗРОБКА ПРОГРАМИ В ГРАФІЧНОМУ РЕДАКТОРІ LD

Ціль роботи: освоєння приймань і методики розробки програм мовою LD у графічному редакторі.

2.1 Методика роботи в графічному редакторі

Графічний редактор забезпечує наступні настроювання.

Масштаб (Zoom).

Розміри графічних елементів у мовах LD, FBD, SFC і CFC можуть бути змінені. За замовчуванням будь-який об'єкт зображується в масштабі 100%. При збереженні проекту коефіцієнт масштабування зберігається. Однак роздруківка проекту на принтер завжди відбувається з масштабом 100%. Установлювати масштаб можна, тільки якщо обрано графічний об'єкт або об'єкт візуалізації.

Ланцюг.

У редакторі LD програма представлена у вигляді списку ланцюгів. Кожний ланцюг складається із двох частин: у лівій записаний номер ланцюга, а в правій – структура, що полягає з логічних або арифметичних операцій, викликів програм, функцій або функціональних блоків, інструкцій переходу або повернення.

Мітка.

Кожний ланцюг може мати мітку, за замовчуванням вона відсутня. Мітку можна поставити, якщо клацнути по першому рядку ланцюга, прямо за номером ланцюга. Після цього можна вводити ім'я мітки, що кінчається двокрапкою.

Коментарі до схеми.

У редакторах релейних і функціональних блокових діаграм будь-який ланцюг (схема) може мати коментар в одну або кілька рядків. Відображення коментарів задаються в опціях (Extras ► Options).

У редакторі релейних схем існує можливість постачити коментарями окремі контакти й обмотки. Для цього необхідно включити опцію Comments per Contact і вставити в поле Lines for Contact Comment число рядків, які потрібно зарезервувати для відображення таких коментарів.

Графічні редактори в режимі Online

У режимі Online можна встановлювати точки останову. Номер ланцюга, у якому встановлена точка останову, зображується синім. Програма зупиняється перед таким ланцюгом, після чого номер схеми стає червоним. Якщо використовуються команди Step in або Step out, то виконується один ланцюг і програма зупиняється.

Позиції курсору в редакторі LD.

Курсор може перебувати в наступних позиціях (рис. 2.1 – 2.3).

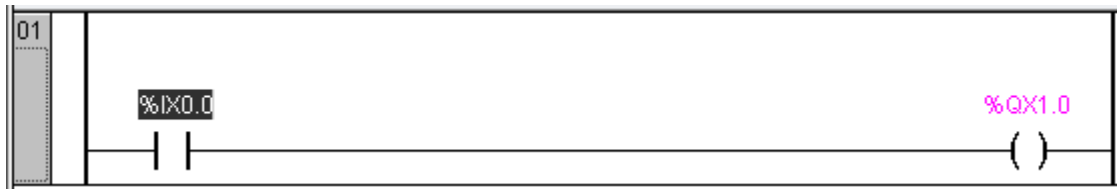


Рисунок 2.1 – Будь-яке текстове поле виділяється чорною рамкою

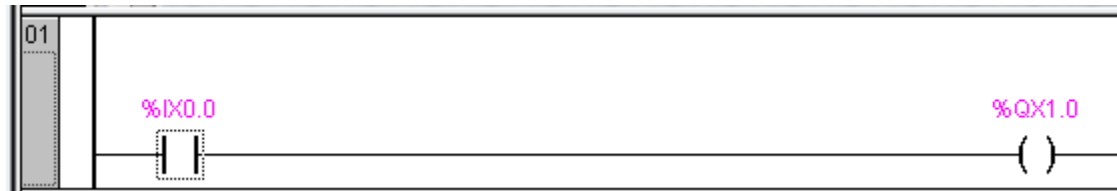


Рисунок 2.2 – Будь-який контакт або обмотка виділяється пунктирною рамкою

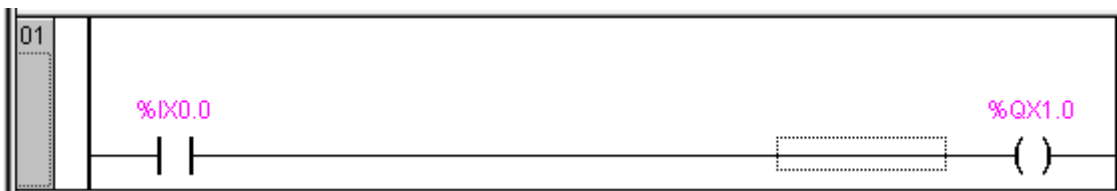


Рисунок 2.3 – Лінія, що з'єднує контакт і обмотку, виділяється пунктирною рамкою

Переміщення елементів і найменувань у редакторі LD.

За допомогою перетаскування мишкою (drag&drop) елементи (контакт, обмотку або функціональний блок) або їх найменування в LD можна переміщати в інші позиції. У процесі цього всі припустимі місця для приміщення елемента будуть показані сірими прямокутниками (рис. 2.4).

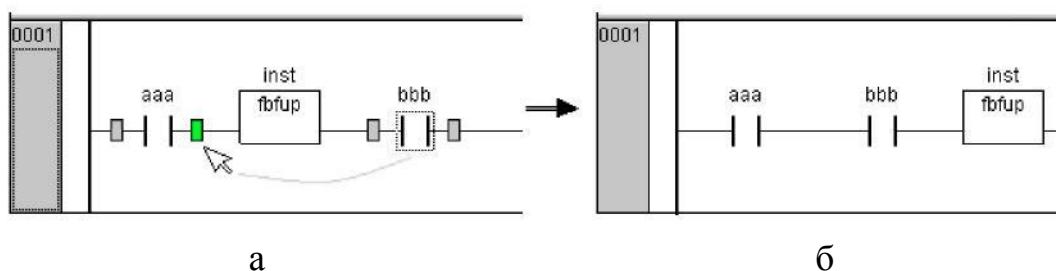


Рисунок 2.4 – Зображення місць можливого приміщення елемента (а) і результат переміщення (б)

Якщо перетягнути елемент у поле імені іншого елемента, то дане поле буде підсвічено зеленим кольором. Якщо тепер відпустити клавішу мишки, то ім'я в полі буде замінено «перетягнутим» іменем (рис. 2.5). Якщо включене відображення адреси й коментаря (опція), то вони також будуть скопійовані.

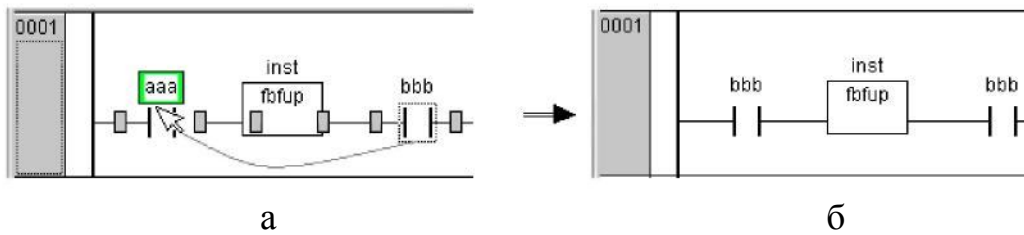


Рисунок 2.5 – Заміна в полі імені (а) і результат заміни (б)

У меню Insert є наступні команди.

Network (before): Команда використовується для вставки ланцюга, вище виділеного.

Network (after): Команда використовується для вставки ланцюга, нижче виділеного.

Contact: Команда використовується для вставки контакту перед обраною позицією в ланцюзі. Текстове поле над контактом заповнюється знаками питання. У цьому полі потрібно ввести змінну або константу. Ім'я змінної зручне вводити за допомогою Input Assistant.

Contact (negated): Команда використовується для вставки інверсного контакту. Вона замінює послідовність команд Insert ► Contact і Extras ► Negate.

Parallel Contact: Команда використовується для вставки контакту, паралельного виділеній позиції схеми.

Parallel Contact (negated): Команда використовується для вставки інверсного контакту паралельно обраному.

Coil: Команда використовується для вставки паралельної котушки.

Для цього потрібно виділити обмотку або лінію, що з'єднує контакти й обмотки, і виконати команду. За замовчуванням змінна, пов'язана з котушкою, одержує ім'я “???”, яке можна замінити на будь-яку константу, змінну або адресу. Для цього зручно використовувати Input Assistant.

‘Set’ coil: Команда використовується для вставки Set-котушки, паралельної обраній.

‘Reset’ coil: Команда використовується для вставки Reset- котушки, паралельної обраній.

Function Block: Ця команда використовується для вставки оператора, функціонального блоку, функції або програми. Для цього потрібно виділити обмотку або лінію з'єднання й виконати команду. Новий блок зазвичай має ім'я AND. Це ім'я можна змінити на будь-яке інше (зручно використовувати Input Assistant).

Перший вхід і перший вихід ROU з'єднуються з деякою лінією зв'язку. Тому вхід і вихід повинні бути типу BOOL. Текстові поля імен змінних для

інших входів і виходів ROU, заповнені трьома знаками питання, потрібно замінити на константи, змінні або адреси.

Box with EN: Команда для вставки ROU з EN-входом. Якщо управляти викликом ROU потрібно з релейного ланцюга, то ROU повинен мати логічний вхід дозволу EN. Вхід EN з'єднується з лінією обмотки, що зв'язує, і контакти. ROU виконується, коли лінія, до якої підключений En-Вхід, передає значення TRUE.

Insert at blocks: За допомогою наявних у цьому списку команд можна вставити новий вхід в ROU (Input), новий вихід в ROU (Output), новий ROU (Box). Команда Assign дозволяє вставити ім'я змінної.

Rising edge detection: Дана команда вставляє в ланцюг функціональний блок R_TRIG, який служить для виділення переднього фронту імпульсу (FALSE -> TRUE) сигналу.

Falling edge detection: Дана команда вставляє в ланцюг функціональний блок F_TRIG, який служить для виділення заднього фронту імпульсу (TRUE -> FALSE) сигналу.

Timer (TON): Дана команда вставляє в ланцюг функціональний блок таймер TON, який часто застосовується для формування затримки сигналу.

Jump: За допомогою цієї команди можна вставити інструкцію переходу, причому ця інструкція розміщується в позиції, що впливає за останньою обмоткою. Якщо лінія, з якою зв'язана інструкція переходу, передає значення On, то здійснюється перехід на зазначену мітку. Виділена позиція повинна бути обмоткою або лінією, що з'єднує обмотку й контакт. Відразу після виконання цієї команди в полі введення імені мітки з'являється рядок “???” для введення імені мітки.

Return: У редакторі LD за допомогою цієї команди можна вставити інструкцію повернення, причому ця інструкція розміщується в позиції, що впливає за останньою обмоткою. Якщо лінія, з якої зв'язана інструкція переходу, передає значення On, то здійснюється перехід на початок виконуваного ROU. Виділена позиція повинна бути котушкою або лінією, що з'єднує обмотку й контакт.

Команди меню Extras.

Paste after: Ця команда використовується для вставки вмісту буфера *за контактом* виділеної позиції. Команда доступна, якщо вміст буфера й виділена позиція – схема, що полягає з контактів.

Paste below: Команда використовується для вставки вмісту буфера *нижче виділеної позиції*. Ця команда доступна, якщо вміст буфера й виділена позиція – схема, що полягає з контактів. Ця схема вставляється паралельно обраній.

Paste above: Команда використовується для вставки вмісту буфера *вище виділеної позиції*. Команда доступна, тільки тоді, коли вміст буфера й виділена

позиція – схема, що полягає з контактів. Ця схема вставляється паралельно обраній.

Negate: Команда використовується для інвертування обраного контакту, обмотки, інструкції переходу або повернення, входу або виходу ROU. При цьому в символі обмотки або контакту з'являється слеш (/) або ||. При інвертуванні інструкції переходу або повернення, входів або виходів ROU з'являється кружок у точці з'єднання.

Інверсна обмотка записує у відповідну логічну змінну значення, зворотне своєму. Інвертований контакт замикає схему, якщо відповідна логічна змінна має значення False.

Set/Reset: Якщо виділити обмотку й виконати цю команду, то можна одержати Set-катушку (позначається буквою “S”). Така катушка записує у відповідну логічну змінну значення True, коли на вході цієї обмотки є сигнал On, і зберігає значення цієї змінної, коли на вході сигнал Off.

Виконавши цю команду ще раз, одержимо Reset-обмотку (позначається буквою “R”). Така обмотка записує у відповідну логічну змінну значення False, коли на вході цієї обмотки є сигнал On, і зберігає значення цієї змінної, коли на вході сигнал Off. Виконавши цю команду ще один раз, одержимо звичайну обмотку.

Команди меню Online.

Команди режиму Online стають доступними тільки тоді, коли буде встановлене з'єднання з контролером.

Login: Команда встановлює з'єднання системи програмування CoDeSys з контролером (або запускає програму емуляції) і включає режим Online. Якщо при цьому проект не відкомпільований, то він компілюється (те ж саме виконує команда Project ► Build). Якщо при компіляції будуть виявлені помилки, то CoDeSys не виконує Login.

Після вдалого з'єднання стануть доступні всі функції Online. Буде здійснюватися моніторинг усіх оголошених змінних.

Logout: З'єднання з контролером розривається або, якщо робота відбувається в режимі емуляції, програма закінчує роботу. Система переходить у режим Offline.

Download: Команда завантажує код проекту в контролер.

Інформація про завантаження зберігається у файлі <ім'я проекту>000000ar.gi, який використовується, якщо система підтримує можливість Online Change (зміни в режимі Online). Цей файл видаляється командою Project ► Clear all. Залежно від цільової платформи при кожному створенні завантажувального проекту *.gi файл може генеруватися заново.

Run: Команда запускає програму в контролері або в режимі емуляції.

Stop: Команда зупиняє програму при її виконанні в контролері або в режимі емуляції.

Reset: Скидання. Команда заново ініціалізує усі змінні, за винятком VAR RETAIN.

Reset (cold): Холодне скидання. Команда виконує ті ж дії, що й команда Reset, але додатково виконує ініціалізацію енергонезалежної області пам'яті RETAIN.

Reset (original): Заводське скидання. Команда ініціалізує PERSISTENT область і видаляє програму користувача. Іншими словами, відновлює стан контролера, у якому він надходить із заводу виготовлювача.

Toggle Breakpoint: Команда встановлює точку останову в поточній позиції активного вікна. Якщо в цій позиції вже стоїть точку останову, то вона буде вилучена. Установити або вилучити точку останову можна, клацнувши по номеру ланцюга мишкою.

Якщо програма була зупинена на точці останову, то позиція точки останову стає червоною. Для того, щоб продовжити виконання програми, слід використовувати команди Run, Step in, або Step Over.

Breakpoint Dialog: Команда відкриває діалог керування точками останову в проекті. У ньому зазначені всі встановлені точки останову.

Для того, щоб установити точку останову, потрібно вибрати в списку POU, що випадає, необхідний POU і номер рядка або ланцюга в списку Location. Точка останову буде встановлена в обраній позиції після натискання кнопки Add. Для видалення точки останову виберіть у списку потрібну точку останову й натисніть кнопку Delete. Кнопка Delete All видаляє всі точки останову. Перехід до позиції точки останову здійснюється так: потрібно вибрати точку й натиснути кнопку Go to.

Step over: Команда виконує одну інструкцію програми. Якщо це інструкція виклику POU, то при виборі цієї команди обраний POU виконується цілком і після цього програма зупиняється. Для того, щоб виконати цей POU по кроках, використовується команда Step in.

Step in: Команда виконує програму по кроках із заходом у викликувані блоки. Викликувані POU відкриваються в окремих вікнах.

Single Cycle: Дану команду можна повторювати багаторазово при відстеженні роботи програми по робочих циклах. Команда виконує один робочий цикл контролера й зупиняється.

Write values: За допомогою цієї команди можна перед початком робочого циклу записати в змінну або в декілька змінних певні значення. Для того щоб поміняти значення логічної змінної, досить двічі клацнути по ній мишкою в розділі оголошень або вікні моніторингу. Діалогове вікно при цьому не з'являється.

Для установки значення не логічної змінної виділіть змінну й натисніть <Enter>. З'явиться діалогове вікно Write variable <x>, у якому потрібно ввести нове значення змінної. Установлене значення виводиться шрифтом бірюзового кольору після поточного значення змінної.

Перш ніж значення змінних будуть записані в контролер, вони зберігаються в списку записуваних змінних (Writelist), поки не будуть внесені в список фіксованих змінних (Forcelist) командою Force values.

Force values: За допомогою цієї команди можна зафіксувати значення однієї або декількох змінних. Запис заданого вами значення здійснюється на початку й наприкінці кожного керуючого циклу.

Release force: Команда скасовує фіксацію змінних. Після виконання цієї команди змінні працюють у програмі як звичайно.

Для того, щоб скасувати фіксацію окремих змінних, їх спочатку потрібно вибрати. Відзначені змінні позначаються словом Release Force бірюзового кольору. Зробити це можна одним зі способів:

- Клацнути по зафіксованій *нелогічній* змінній у вікні монітора й у діалозі, що з'явився, Write variable <x> натиснути кнопку Release Force for this variable.
- Клацати по зафіксованій *логічній* змінній до появи напису Release Force.
- У меню Online за допомогою команди Write/Force-Dialog відкрити діалог і вилучити значення змінної в стовпці Forced value.

Коли всі необхідні змінні відзначені Release Force у вікні оголошення, потрібно використовувати команду Force values для передачі змін у контролер.

Write/Force Dialog: Якщо під час виконання команди Release Force список Writelist не порожній, то буде відкритий діалог Remove Write-/Forcelist. У ньому можна вказати, який список видаляти – або Writelist (Remove writelist), або Forcelist (Release force). Можна також вилучити обидва списки.

Show Call Stack: Команда виводить стек викликів POU і призначена для організації діалогу в режимі Online зі списком POU у точці останову. На першому місці в цьому списку завжди стоїть POU PLC_PRG, тому що виконання програми завжди починається з нього. На останньому місці стоїть POU, у якому програма була зупинена. Якщо вибрати POU і натиснути кнопку Go to, то цей POU буде відкритий у редакторі на поточній команді.

Display Flow control: Включення режиму контролю потоку виконання відображується галочкою перед командою Display Flow control. Якщо дана можливість підтримується в цільовій платформі, то кожний рядок або ланцюг програми, який був виконаний в контролері в попередньому керуючому циклі, буде виділено.

Simulation Mode: Команда включає режим емуляції. Якщо режим емуляції включений, то команда Simulation Mode у меню відзначена галочкою.

У режимі емуляції програма виконується в ПК. Цей режим використовується для тестування проекту. Взаємодія з емулятором опирається на механізм повідомлень Windows. Якщо режим емуляції виключений, то програма буде запущена в контролері. Обмін даними між ПК і ПЛК зазвичай здійснюється по послідовному інтерфейсу.

Слід урахувати дві обставини:

1. РОУ із зовнішніх бібліотек не виконуються в режимі емуляції.
2. Тривалість робочого циклу в режимі емуляції не відповідає (як правило, суттєво більше) тривалості циклу в реальному контролері.

2.2 Приклад створення програми мовою LD

Для засвоєння методики програмування розглянемо приклад створення програми релейного автомата.

Завдання. Потрібно розробити програму кодового замка, який реалізований на електромагнітних реле. Принципова електрична схема кодового замка наведена на рисунку 2.6.

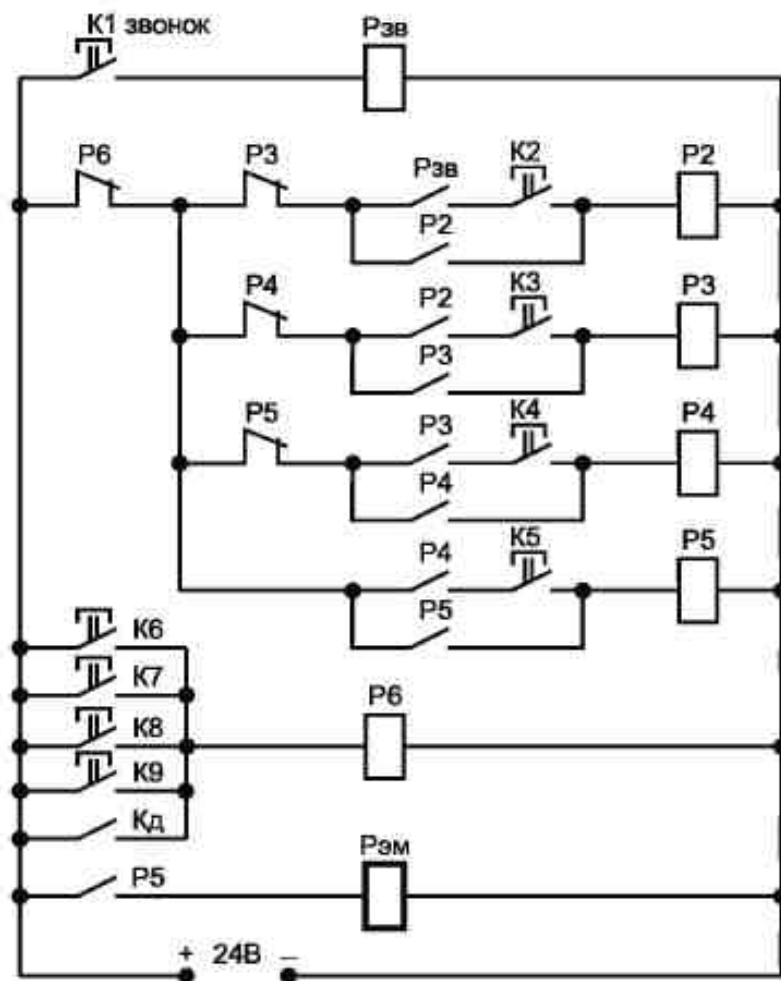


Рисунок 2.6 – Принципова електрична схема кодового замка на електромагнітних реле

Замок постачений складальною панеллю з 10 кнопками. Із них п'ять кнопок (K1-K5) задають код відкривання шляхом послідовного натискання, однак кнопка K1 (дверний дзвінок) і кнопка K2 (перша цифра коду) повинні бути натиснуті одночасно. Усі проміжні реле (P2–P5) працюють із самофіксацією, одночасно звільняючи реле попереднього ланцюга. Помилкове натискання кнопок із групи K6-K9 або натискання кнопки відкривання двері (K_д) скидають замок у вихідний стан.

Таємність цифр коду досягається розпаюванням контактів K2-K5 до різних клавш складального поля. Силкові ланцюги дзвінка голосного бою й електромагнітного плунжера на схемі не показані.

Таким чином, якщо код набраний правильно, то замок відкривається з одним дуже коротким дзвінком, а при спробах добору коду виникає кілька дзвінків. Одночасне натискання всіх кнопок K2-K9 викликає скидання автомата у вихідний стан.

LD-діаграму побудуємо без переробки алгоритму роботи пристрою.

Для реалізації кодового замка на контролері необхідний ПЛК, що має 10 дискретних входів (кнопки K1- K9 плюс датчик відкритих дверей K_д) і два виходи – один для подачі сигналу на дзвінок і інший для включення електромагнітного плунжера.

Послідовність створення проекту.

Новий проект створюється командою File ► New. При цьому виводиться діалогове вікно *Target Settings*. Тому що проект є машино-незалежним, його можна випробувати в режимі емуляції (варіант None).

Наступне діалогове вікно (New POU) визначає тип першого програмного компонента. Тут вибираємо мову реалізації LD і зберігаємо запропоновані за замовчуванням тип (PROGRAM) і ім'я (PLC-PRG) програмного компонента. Після цього CoDeSys установлює панель інструментів і робоче вікно редактора LD.

Оголошення змінних.

Враховуючи те, що вхідні й вихідні сигнали автомата очевидні, то у вікні оголошення змінних після імені програми вставляємо заголовок VAR_INPUT, перераховуємо вхідні змінні (усі кнопки) і привласнюємо їм (за допомогою асистента введення) тип BOOL. Оголошення вхідних змінних завершуємо декларацією END_VAR. Аналогічно поступаємо з вихідними змінними (VAR_OUTPUT). Локальні змінні можна буде повідомляти в процесі побудові ланцюгів програми.

Вікно оголошення змінних показано на рисунку 2.7. Тут кнопки відкривання двері привласнені ім'я Door_open, а дзвінку – Alarm.

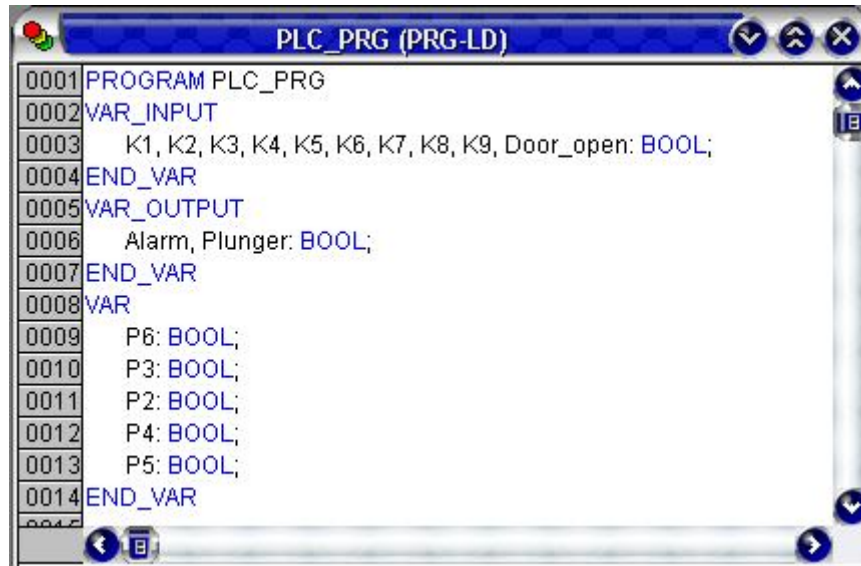


Рисунок 2.7 – Вікно оголошення змінних

Побудова LD-діаграми.

Перший ланцюг містить тільки кнопку дзвінка й котушку дзвінка. Вибираємо на панелі інструментів нормально розімкнутий контакт – він автоматично вставляється в лінію ланцюга подвійним кліком.

Три питання в полі оголошення імені контакту замінюємо за допомогою асистента введення (F2) на K1. Далі вибираємо котушку (Coil) і вводимо її ім'я – Alarm.

Другий ланцюг доцільно розділити – за правилами побудови LD-діаграм у ланцюзі повинна бути тільки одна котушка (обмотка) реле.

Застосовуємо команду контекстного меню Network (after), відповідно до якої (нижче першої) створюється новий ланцюг. У цьому ланцюзі послідовно вставляємо (із присвоєнням відповідних імен) контакти: нормально-замкнені P6, P3, нормально розімкнуті Alarm і паралельний контакт P2. Далі вводимо в ланцюг контакт K2, який перетягуємо усередину паралельного з'єднання. Завершуємо формування ланцюга вставкою обмотки P2. Її ім'я вводимо за допомогою асистента (F2).

Новий ланцюг 0003 створюємо шляхом копіювання ланцюга 0002. Після вставки ланцюга робимо заміну імен елементів у відповідності зі схемою замка. Аналогічно поступаємо з побудовою ланцюга 0004.

Інші ланцюги будуються відповідно до принципової схеми замка. Діаграма LD, що реалізує логіку керування, показана на рис. 2.8 і 2.9.

Легко помітити, що діаграма практично повторює принципову схему. Єдина відмінність полягає в тому, що контакти реле P6 розділені на кілька ланцюгів (ланцюги 2-5). Це диктується правилами побудови діаграм і не впливає на роботу схеми.

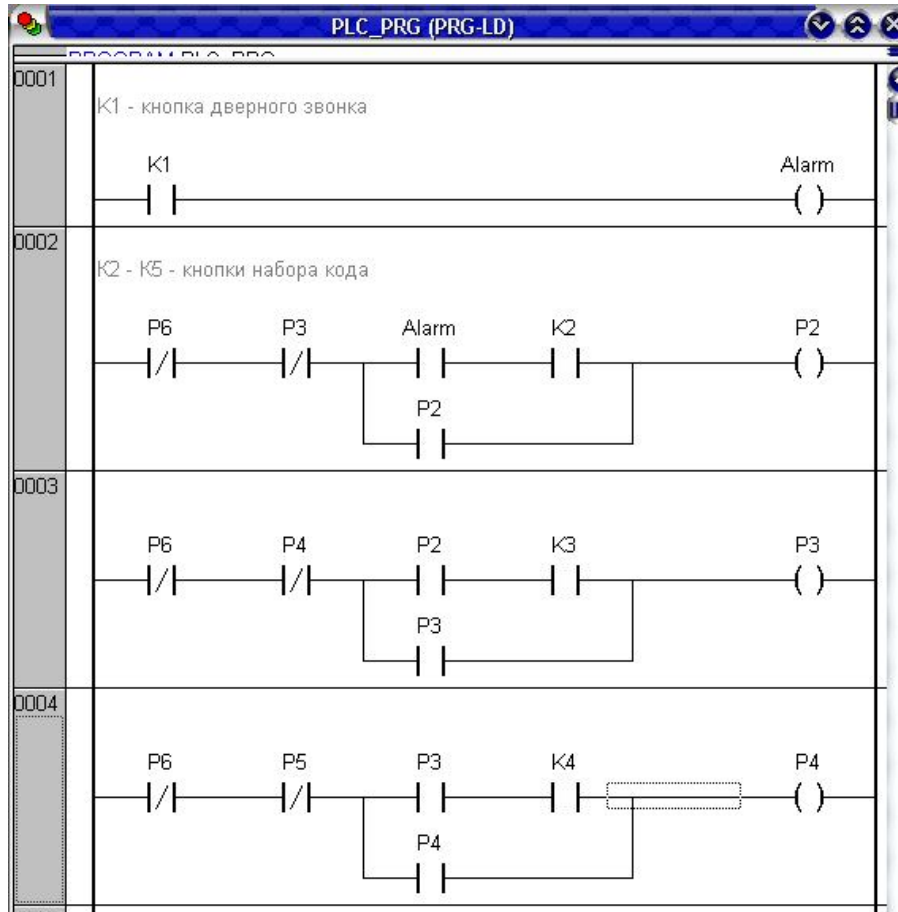


Рисунок 2.8 – LD-діаграма кодового замка (ланцюги 1-4)

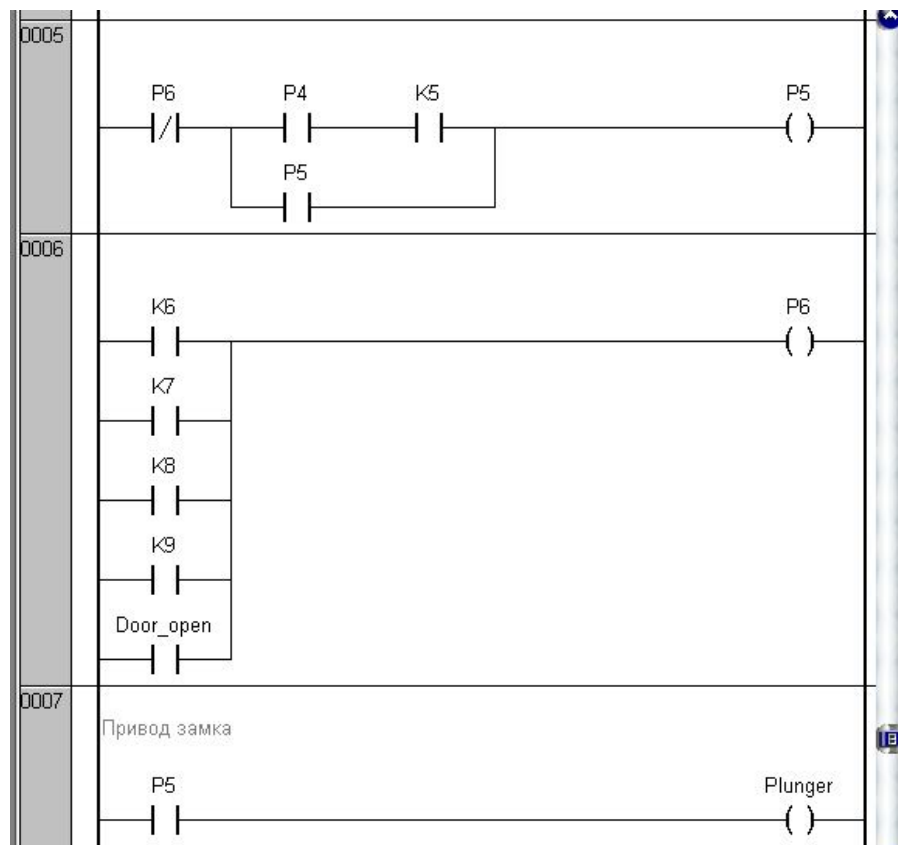


Рисунок 2.9 – LD-діаграма кодового замка (ланцюги 5-7)

Робота з відладчиком.

Для перевірки правильності роботи програми потрібно перейти в середовище віртуального контролера – Online ► Login. При цьому редактор відображає значення змінних і стан кожного ланцюга. Подвійним кліком по кожній змінній K1-K5 змінюємо їх стани з FALSE на TRUE і командою Online ► Write Values записуємо ці зміни у віртуальний контролер.

Після цього запускаємо програму командою Online ► Run.

Результат виконання програми відображається станом ланцюгів – включені контакти (стан On), усі лінії, що передають стан On, і обмотки показуються синім кольором. Крім цього вказуються значення всіх входів і виходів (рис. 2.10).

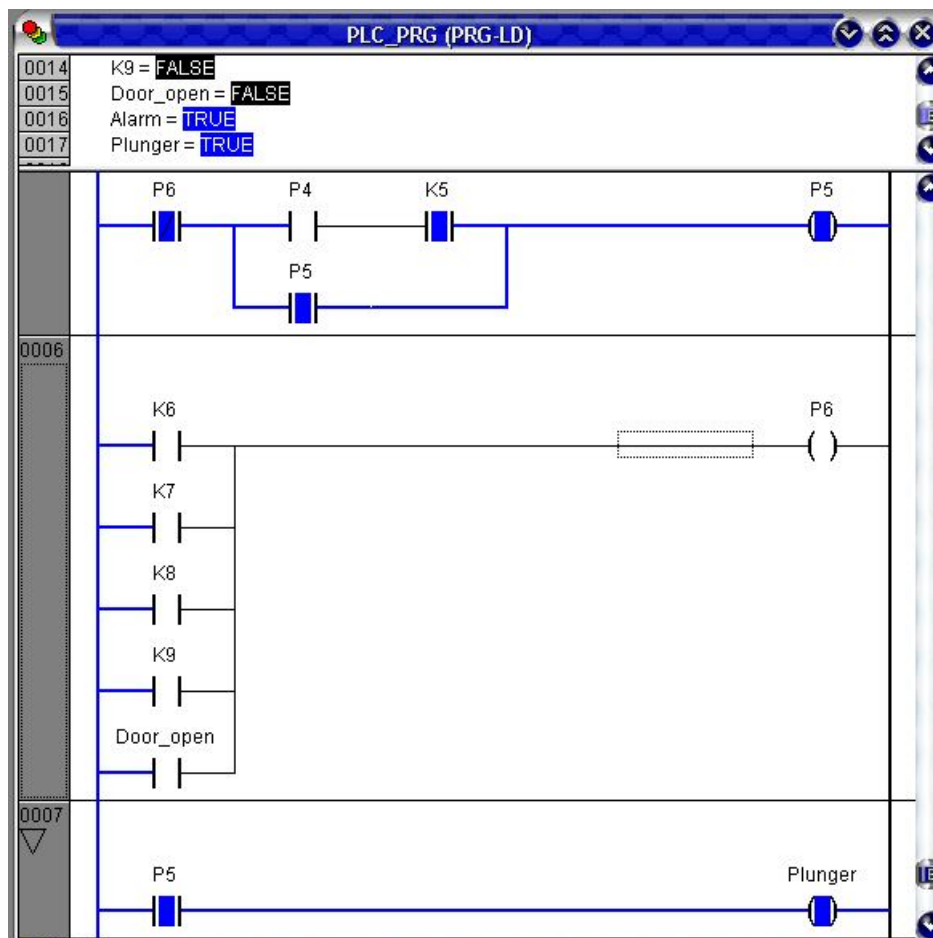


Рисунок 2.10 – Результат виконання програми (ланцюги 5-7) у режимі Online

2.3 Варіанти завдань і зміст звіту по роботі

Варіанти завдань видаються викладачем. Змістом роботи є створення LD-діаграми за рележно-контактною схемою.

Звіт по роботі повинен містити умови завдання (рележно-контактна схема і її опис), а також програмну реалізацію у вигляді LD-діаграми.

Звіт захищається в процесі демонстрації роботи програми.

3 ЕЛЕКТРОТЕХНІЧНЕ ПРОЕКТУВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ В EPLAN ELECTRIC P8

Ціль роботи: освоїти приймання й придбати навички створення проектної документації по системі автоматизації за допомогою програми EPLAN Electric Professional P8.

3.1 Призначення програмного пакета EPLAN

У теперішній час підприємства зустрічаються з необхідністю розробки великого обсягу конструкторської документації в короткий термін. У зв'язку із цим потрібно автоматизувати основні процеси проектування за допомогою спеціалізованої САПР. Для розв'язку цього завдання слід застосовувати пакет EPLAN.

Програмний пакет EPLAN – це комплекс програмних засобів для розробки проектної документації по системах автоматизації.

У кожного завдання проектування є свої вимоги до інструменту для її розв'язку. Тому пакет EPLAN містить у собі ряд програм, основними з яких являються: EPLAN PPE, EPLAN Electric P8 Professional, EPLAN Cabinet і EPLAN Fluid.

Система EPLAN PPE спрямована на створення технологічних схем, проектування пристроїв контурів виміру й керування, а також підготовку кошторисної документації.

EPLAN PPE дозволяє інтегрувати дані в САЕ-систему EPLAN Electric P8, призначену для автоматизованого створення принципів схем, а також схем з'єднань і підключень.

Програма EPLAN Cabinet призначена для проектування електричних шаф керування на основі проектної документації, створеної в EPLAN Electric P8.

Програма EPLAN Fluid призначена для гідротехнічного інжинірингу з автоматизованим обліком документації. Ця програма може використовуватися в якості доповнення до САЕ системи EPLAN Electric P8.

Ціль даного посібника – дати рекомендації студентам по освоєнню програмної системи EPLAN Electric P8 для проектування електричних схем при виконанні практикуму по дисципліні «Основи проектування систем автоматизації», а також при виконанні курсових і дипломних проектів.

До основних переваг EPLAN можна віднести наступне.

1. EPLAN орієнтований на розробку систем автоматизації виробничих процесів. Він дозволяє розробляти проекти в стислий термін і з високою

якістю. У пакеті EPLAN зображення графічних символів пристроїв, нумерація проводів на схемах, складання таблиць проводів для монтажу, створення форм документів, внесення в документи перехресних посилань виконується автоматично. При цьому виключаються часто виникаючі при ручнім проектуванні помилки, такі як неправильні номери виводів елементів схем, дубльовані номери проводів, пропуски в нумерації проводів і елементів, пропуски або невірні перехресні посилання й т.п.

2. EPLAN дозволяє організувати спільну роботу над документами проекту групою розроблювачів, звільняючи інженерний склад від рутинної роботи.

3. База даних EPLAN може містити як стандартні елементи виробників Schneider Electric, ABB, Siemens і інших, так і власні, створювані користувачем у процесі проектування.

4. У базі й документах, що генеруються EPLAN, технічна інформація легко зв'язується з комерційною, що дозволяє одержувати специфікації з вартістю комплектуючих компонентів і всієї АСУТП у цілому.

5. Усі дані по входам і виходам промислових контролерів, використовуваних у розроблювальній АСУТП, можуть бути зведені в один документ, незалежно від того, у яку частину проекту ці дані були внесені.

6. Документація, розроблена в EPLAN, може бути представлена як окремими документами, так і у вигляді єдиного комплекту. Мова й формат представлення документації вибираються користувачем.

7. Кількість робочих місць може розширюватися в міру необхідності. Пакет може нарощуватися за рахунок додатка, що дозволяє управляти проектом у цілому, додатка, що дозволяє виконувати компонування встаткування і т.д.

8. Документи проекту, виконаного в EPLAN, можуть перетворюватися в інші формати, зокрема, HTML і PDF. При цьому зберігаються перехресні посилання усередині проекту. Можливе перетворення у формати інших розроблювачів, зокрема, пакета Autocad (файли .dwg) і інших розповсюджених пакетів САПР. Можливий імпорт/експорт як цілих документів, так і окремих графічних символів.

До недоліків EPLAN можна віднести:

1. Низький рівень підтримки пакета EPLAN в Україні, що утрудняє його освоєння.

2. Пакет споконвічно розроблявся для західного користувача й не завжди дозволяє адаптувати проекти до вимог нормативної документації України.

3. Мінімально припустимі вимоги до продуктивності комп'ютера, що декларуються виробником, явно занижені. Для нормальної роботи потрібно більш могутніший комп'ютер.

4. Бази даних, що поставляються в комплекті з пакетом комплектуючих від різних виробників, задовольняють вимоги розроблювачів лише частково. Для реальної роботи доводиться робити доповнення баз даних і створювати власні.

3.2 Завдання проектування й рекомендації зі створення проекту

Електротехнічне проектування системи автоматизації в рамках комп'ютерного практикуму передбачає розв'язок наступних завдань:

- 1) створення структури проекту;
- 2) створення *принципової схеми* електроживлення системи автоматизації (головному електричному кола);
- 3) створення *схем підключення* вхідних і вихідних пристроїв до модулів програмувального логічного контролера;
- 4) створення *схеми з'єднань* засобів керування й аварійного захисту периферійного встаткування системи автоматизації;
- 5) створення *звіту* по проекту (специфікацій та таблиць з'єднань).

Для виконання цих завдань студент одержує індивідуальне завдання (зразки схем). Суть роботи зводиться до того, що студент, використовуючи видані йому зразки, виконує перераховані вище схеми в графічному редакторі САЕ-системи EPLAN Electric Professional P8, як єдиний проект. Звіт по виконаній роботі створюється в EPLAN автоматично.

При запуску EPLAN відкривається вікно «Вибрати ліцензію» (рис. 3.1), у лівім полі якого (**Выбор**) позначені доступні програми, що мають ліцензії. Програми, позначені червоним кружком із хрестиком, ліцензії не мають і тому недоступні.

З доступних ліцензій вибираємо ліцензію на EPLAN Electric P8 Professional і закриваємо вікно кнопкою ОК.

При запуску програми виводиться діалогове вікно вибору категорії користувача, у яким слід зазначити «Просунутий користувач» або «Експерт».

Для створення нового проекту необхідно прийняти шаблон проекту, за допомогою якого створюються необхідні налаштування проекту. У шаблоні проекту втримується структура позначень сторінок і пристроїв. Файли для шаблонів проектів мають розширення **.ept**. Процес створення проекту суттєво спрощується за допомогою майстера проекту.

Рекомендується наступна послідовність створення проекту.

1. Виберіть пункт меню **Проект > Создать (мастер)**. Відкриється діалогове вікно **Создать проект**.

2. На першій вкладці в поле **Имя проекта** введіть назву проекту (можна російською мовою).

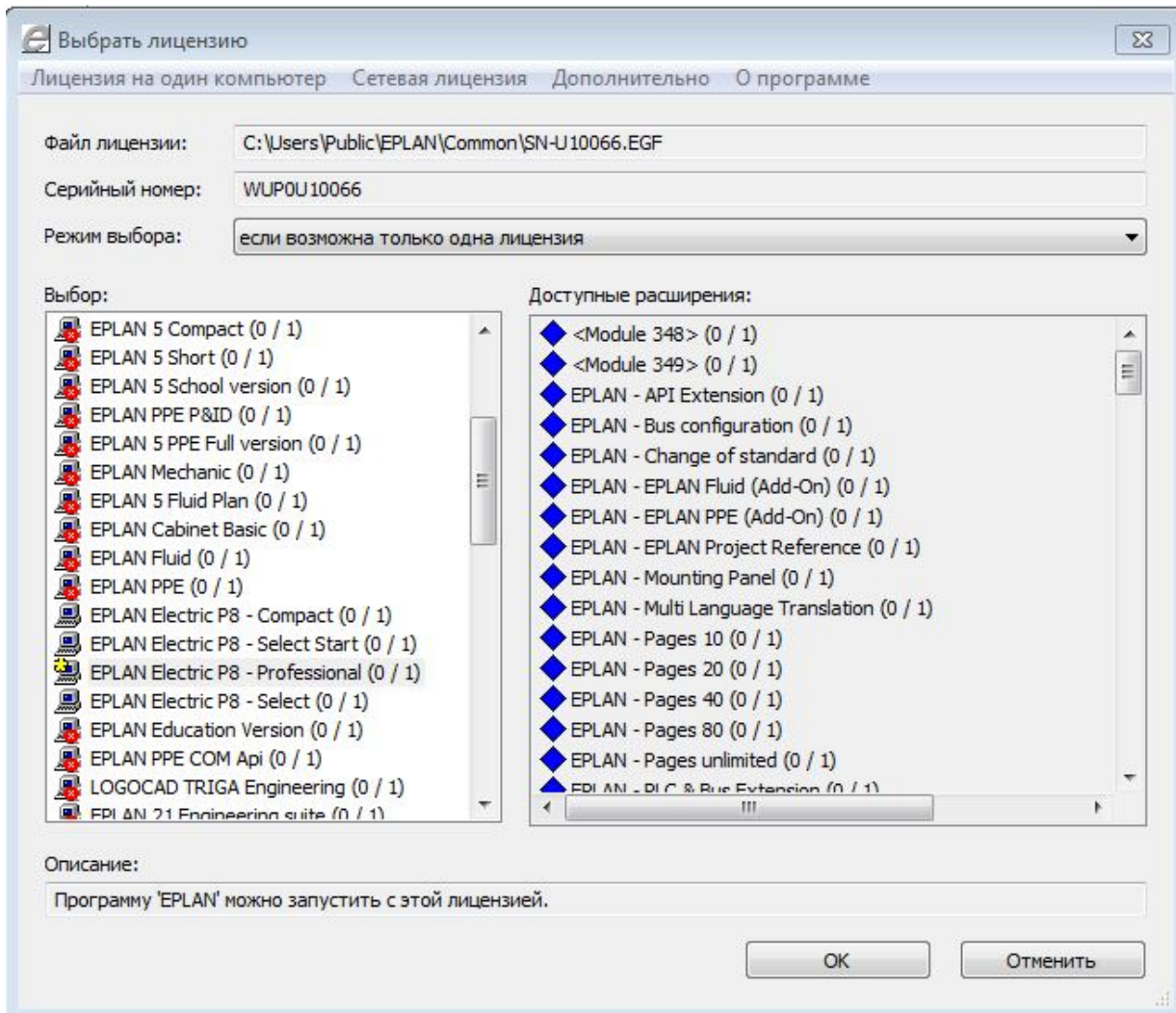


Рисунок 3.1 – Вікно вибору ліцензованої програми

3. Виберіть **Шаблон** проекту. Для цього клацніть по кнопці [...], яка перебуває поруч із однойменним полем. Відкриється діалогове вікно **Открыть**.

4. Виділіть у діалоговому вікні **Открыть** шаблон проекту GOST_tpl001.ept.

5. Клацніть по кнопці **Открыть**. Діалогове вікно **Открыть** буде закрито, а назва шаблону проекту буде скопійована.

6. Виберіть для проекту **Место хранения**. Для цього клацніть поруч із однойменним полем по кнопці [...]. Відкриється діалогове вікно **Обзор папок** (рис. 3.2).

7. За замовчуванням проекти зберігаються в каталозі **Проекты**, у створеній під час інсталяції папці фірми, наприклад, «APP». Виділіть папку **Проекты** й клацніть по кнопці **Создать папку**. У середині виділеної папки буде створена нова папка, якій потрібно привласнити ім'я проекту. Після цього закрийте вікно кнопкою ОК.

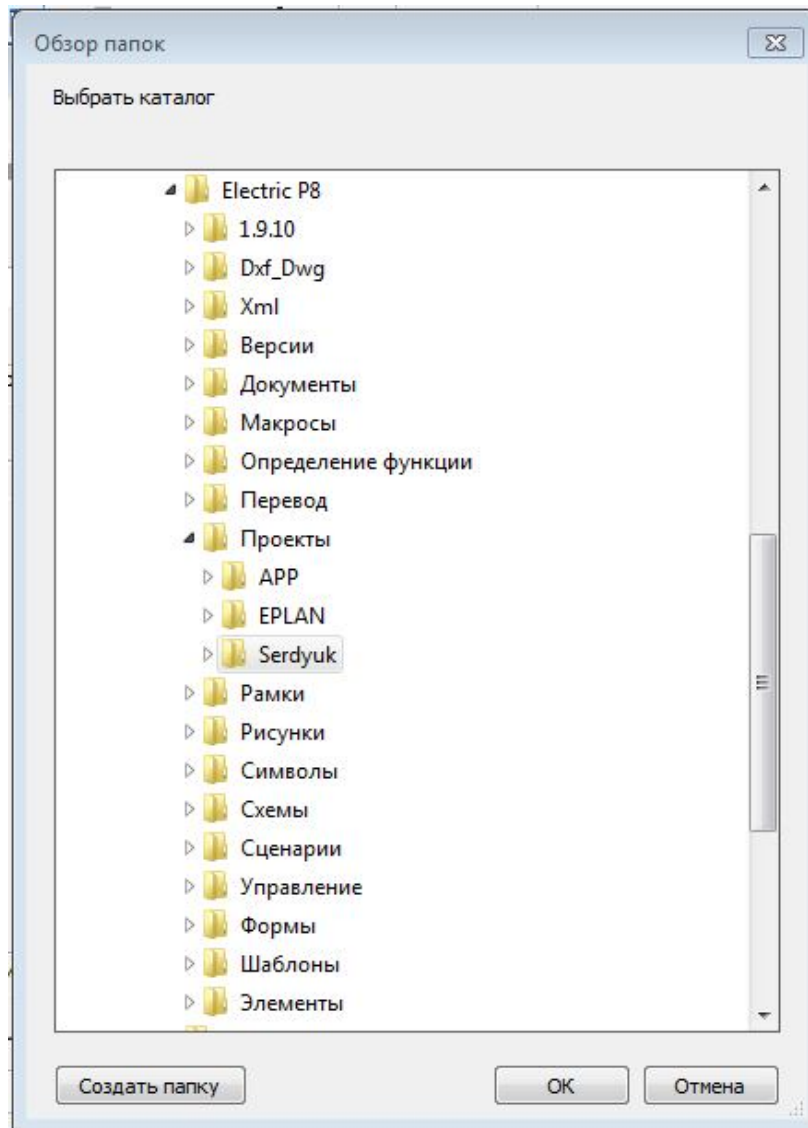


Рисунок 3.2 – Діалогове вікно *Обзор папок* програми *EPLAN Electric*

8. У вікні **Мастер проекта** клацніть по кнопці **Завершить**. Відкриється діалогове вікно **Импортировать проект**. Майстер проектів скопіює шаблон у новий проект. Ця операція може протривати якийсь час. Після цього діалогове вікно **Создать проект** слід закрити й у **Навигаторе страниц** з'явиться створений проект.

Проект в EPLAN складається з колекції різних типів документів – титульного аркуша, схем, відомостей, таблиць і т.п. Усі документи компонуються в розділи проекту по типам сторінок. Розділам відповідають різні піктограми в **Навигаторе страниц**.

Схеми з'єднань створюються усередині розділів. Усі об'єкти проекту (розділи, сторінки, пристрої й функції) повинні бути позначені ідентифікаторами й повинні утворювати в рамках проекту ієрархічну структуру, яка дозволяє швидко знайти потрібну схему або окремих пристрій. Позначення для структурування проекту називають "Структурними ідентифікаторами".

При створенні сторінки їй привласнюється певний тип, який згодом може бути змінений.

При створенні проекту його структуру можна зробити з мінімальною кількістю ідентифікаторів, як показано на рисунку 3.3. Це дозволить спростити роботу з рамками схем. Якщо для всіх схем застосувати один ідентифікатор місця, то перша схема буде мати основний напис за формою 1, інші схеми – за формою 2а. Для титульного аркуша основний напис повинен мати форму 2.

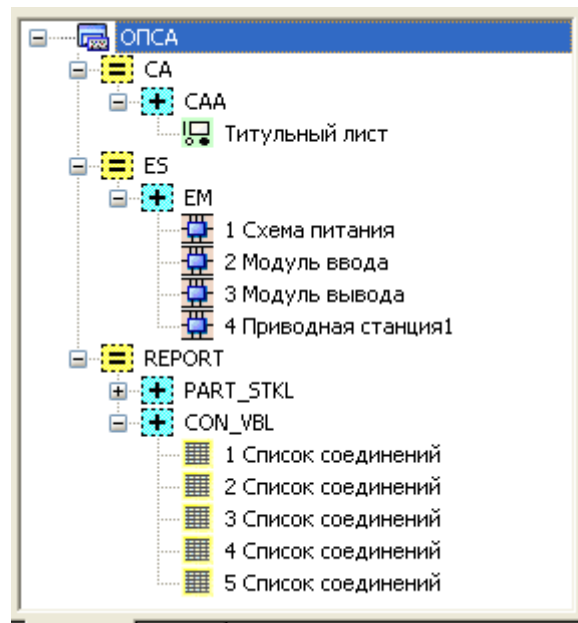


Рисунок 3.3 – Рекомендована структура проекту

На рисунку 3.3 ідентифікатори призначені з наступних міркувань.

Титульний аркуш – це *текстовий* документ, який виконується на сторінці форматом А4 з основним написом для текстового документа. Тому для його ідентифікації застосований покажчик на розділ текстової документації СА (система автоматизації), і покажчик на тип документа – САА.

Схеми є іншим типом документів. Вони виконуються на аркушах формату А3 і забезпечуються основним написом *для креслень*. У зв'язку із цим для ідентифікації комплекту схем призначені покажчик розділу схем ES і покажчик підрозділу устаткування EM (рис. 3.3).

Для звітної документації ідентифікатори не призначаються – вони генеруються разом зі звітом автоматично.

При створенні сторінки титульного аркуша слід вибрати рамку GOST_A4_first_page_text. Перш ніж використовувати цю рамку, її необхідно відкрити й відредагувати. Для цього виділіть титульний аркуш у навігаторові й виконайте: **Сервисные программы ► Основные данные ► Рамка ► Открыть** (можна застосувати праву кнопку й вибрати команду **Рамка ► Открыть**). У вікні **Открыть рамку** виберіть зі списку необхідний файл і **натисніть** ОК. У навігаторові під іменем проекту з'явиться сторінка з

позначенням рамки «GOST...». Одночасно з'явиться відображення рамки в робочій області редактора.

У вікні редактора потрібно вилучити з рамки зайві тексти, наявні в основному написі, і закрити рамку командою **Сервисные программы ► Основные данные ► Рамка ► Закреть**. При цьому у вікні редактора з'явиться повідомлення про зміну основних даних і запит на відновлення даних відкритого проекту. Підтвердіть запит кнопкою **Да**. З'явиться вікно відображення динаміки процесу копіювання й по його завершенню сторінка з рамкою під іменем проекту зникне, а титульний аркуш буде постачений новою рамкою.

Усі наступні текстові записи (найменування проекту, номер документа, прізвища й ін.) необхідно вставляти *в рамку* при роботі зі сторінкою, *не відкриваючи рамку*. Відкривати рамку слід тільки при її заміні.

При створенні схем потрібно застосовувати наступні рамки:

- GOST_first_page_scheme_A3 – для першої сторінки;
- GOST_next_page_scheme_A3 – для наступних сторінок.

Створення сторінок проекту

Клацніть правою кнопкою на імені проекту. У контекстному меню виберіть операцію **Создать**. Відкриється діалогове вікно **Новая страница**, у якій є поле **Полное имя страницы**.

За замовчуванням в ім'я включені символи структурних ідентифікаторів проекту (= і +) а також розділова похила риса, після якої слідує найменування схеми. При цьому знак «=» є *попереднім* знаком для наступного за ним структурного ідентифікатора блоку ідентифікаторів "Установка", а знак «+» є *попереднім* знаком для наступного за ним структурного ідентифікатора блоку ідентифікаторів «Місце установки».

Для створення повного імені сторінки необхідно натиснути кнопку [...], яка перебуває поруч із однойменним полем. EPLAN відкриє діалогове вікно **Полное имя страницы**, у якому потрібно виконати наступне.

1. У рядку знака структурного ідентифікатора "=" увести коротке ім'я верхнього рівня ієрархії документації, наприклад, ES (Electrotechnical schemes – електротехнічні схеми).

2. У рядку знака «+» увести коротке ім'я наступного рівня ієрархії, наприклад, EP (Електросхеми живлення).

3. У поле імені сторінки слід увести найменування схеми, наприклад, «Схема живлення». Виконані призначення показані на рис. 3.4.

Закриваємо вікно кнопкою ОК і вертаємося у вікно **Новая страница**. Повне ім'я буде відображено з іменами ідентифікаторів.

Наступним полем вікна **Новая страница** є **Тип страницы**. Для побудови схем з'єднань слід застосувати тип «Многополюсная схема соединений».

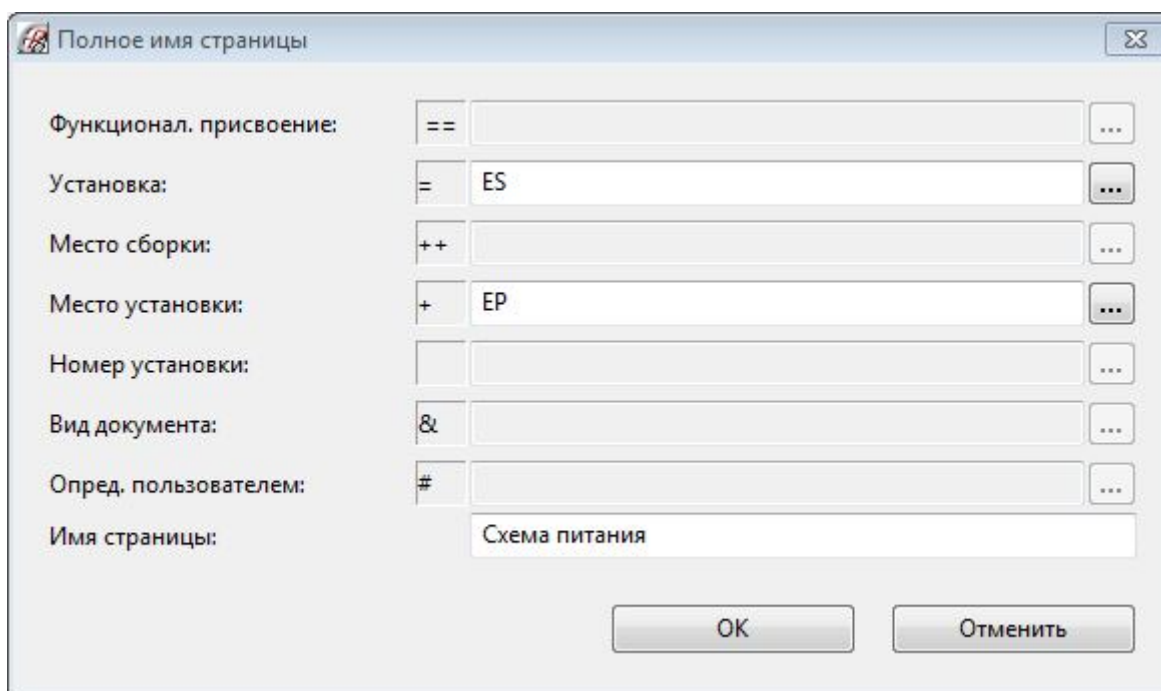


Рисунок 3.4 – Вид вікна призначення повного імені сторінки

Багатополіусна схема з'єднань – це схема, що містить багатополіусні символи. На такій сторінці є спеціальні можливості обробки, наприклад, вставлені символи автоматично зв'язуються один з одним.

Далі переходимо до призначення властивостей сторінки. Тут необхідно вибрати рамку креслення. На перетинанні рядка **Имя рамки** й стовпця **Значение** лівою кнопкою миші розкриваємо схований список доступних рамок. Для першого аркуша схеми слід вибрати рамку GOST_first_page_scheme_A3, для наступних сторінок потрібна інша рамка – GOST_next_page_scheme_A3. Усі виконані призначення показано на рис. 3.5.

Після цього закриваємо вікно **Новая страница** кнопкою ОК.

При закритті вікна відкривається наступне вікно – **Позиционировать идентификаторы**. За допомогою цього діалогового вікна визначається послідовність недавно створених структурних ідентифікаторів у рамках одного проекту.

На рисунку 3.6 показаний приклад дерева проекту, створюваного в рамках практикуму по дисципліні ОПСА. Тут передбачені: структурний ідентифікатор установки ES (електричні схеми), а також структурні ідентифікатори місць установки – EP (електроживлення установки), PLC (система керування установки) і EM (електромотори установки і їх керування). Під кожним ідентифікатором місця установки перебувають конкретні електричні схеми.

Створивши сторінки проекту й постачивши їх рамками, слід перейти до створення схем. Для вставки елементів схеми краще використовувати команду **Вставить ► Символ** або **Вставить ► Макрос символа**.

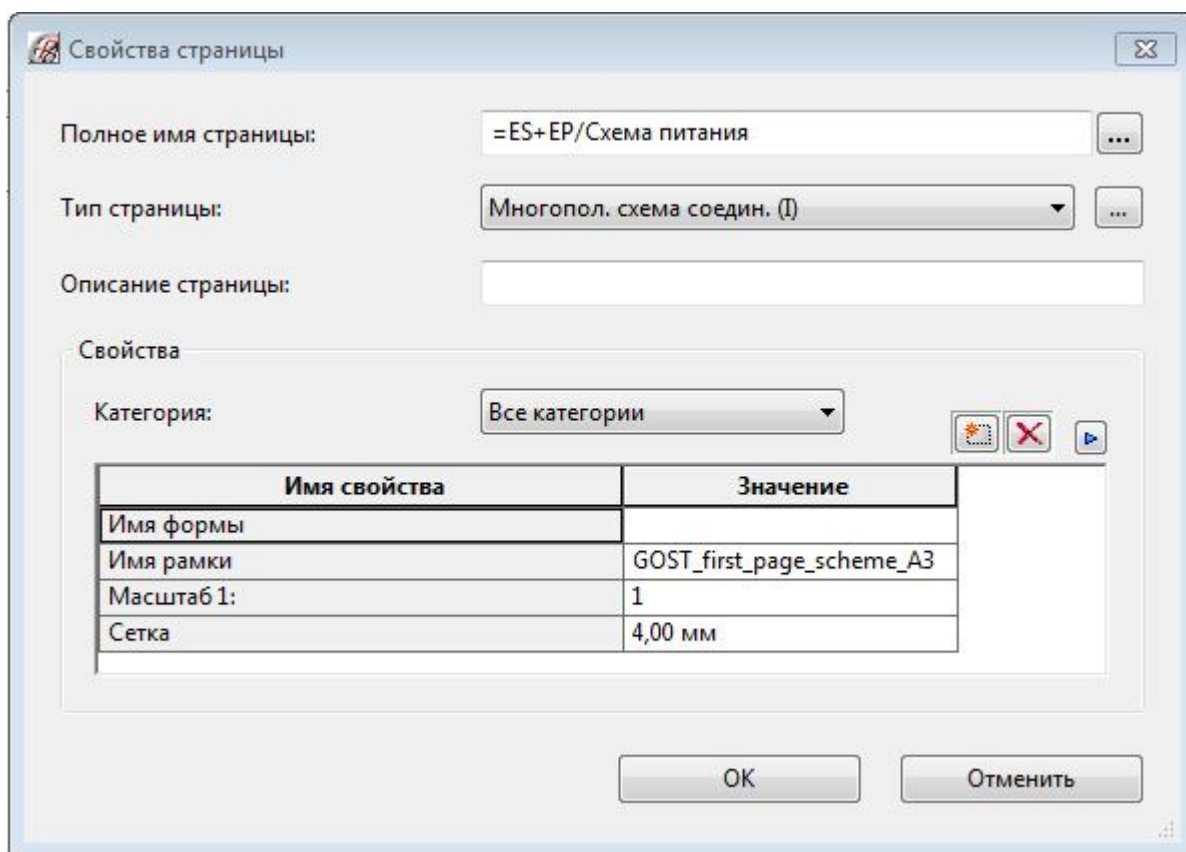


Рисунок 3.5 – Вид диалогового окна **Свойства страницы** з повним іменем сторінки

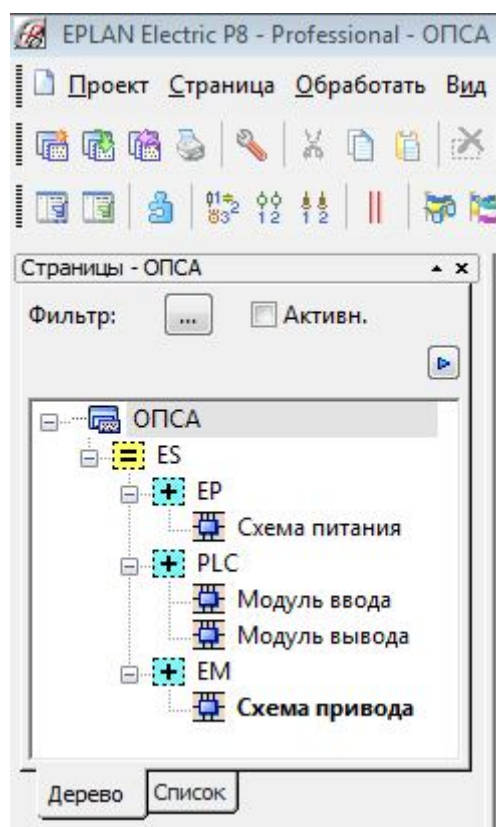


Рисунок 3.6 – Вид дерева проекта в навігаторі сторінок

При вставці символу відкривається вікно **Свойства**, у якому слід спочатку визначити позначення елемента і його технічні характеристики (вкладка «Имя символа»), а потім перейти на вкладку «Изделие» і натиснути кнопку «Выбор устройства». При такій технології вибір пристрою проводиться редактором автоматично, що спрощує цей процес. Якщо заданим на попередній вкладці параметрам задовольняє кілька пристроїв, то для вибору можна врахувати інші параметри, наприклад, виробника.

Макроси символів використовуються для вставки, головним чином, складних елементів або частин схеми, наприклад, модулів контролера або частотних перетворювачів. У папці «Макросы» зібрана велика бібліотека символів для систем автоматизації – бази SIEMENS, Schneider Electric, ABB, OMRON, MOELLER, SEW-EURODRIVE і ін.

Для налаштування інтерфейсу редактора необхідно вибрати **Вид ► Рабочая область**. Відкриється діалогове вікно, у якому можна вибрати схему відображення вікна редагування (рис. 3.7). Для створення принципових електричних схем, а також підключень і з'єднань слід вибрати схеми відображення «Стандарт» або «EPLAN 21».

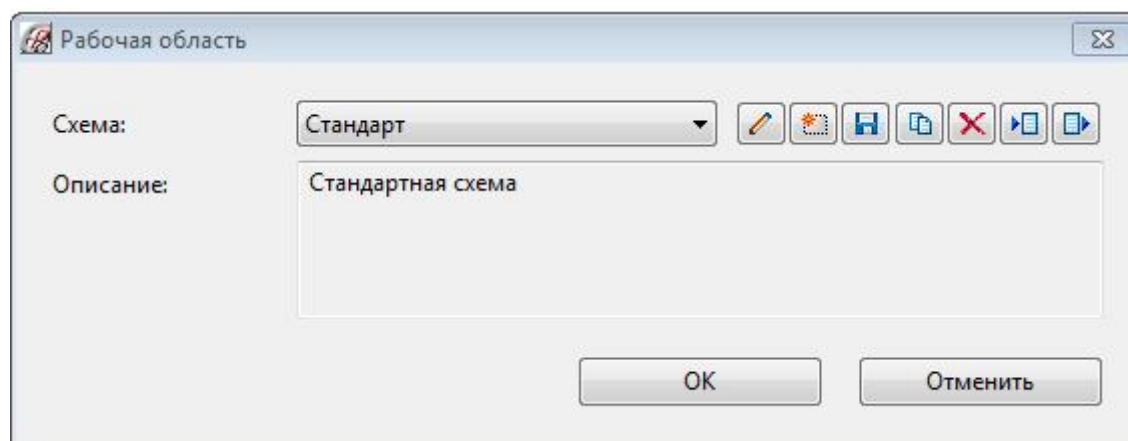


Рисунок 3.7 – Вікно вибору схеми відображення робочої області

3.3 Створення принципових схем і схем підключень

При відкритті сторінки з'являється вікно з робочою областю, у якій відображається бланк креслення з основним написом за ДСТ.

Спочатку слід зробити налаштування відображення сітки й установити опцію **Захват объекта**. Обидві ці опції створюють можливість вирівнювання елементів при малюванні по точках сітки й точках з'єднання елементів. Розмір використовуваної сітки зберігається як властивість сторінки й відображається в рядку стану.

Для налаштування відображення сітки й установки опцій можна використовувати кнопки панелі інструментів, відзначені маркером на рис. 3.8.

За допомогою цих кнопок можна відобразити сітку, настроїти крок (А, В, С, D, Е), а також включити захват сітки й захват об'єкта.

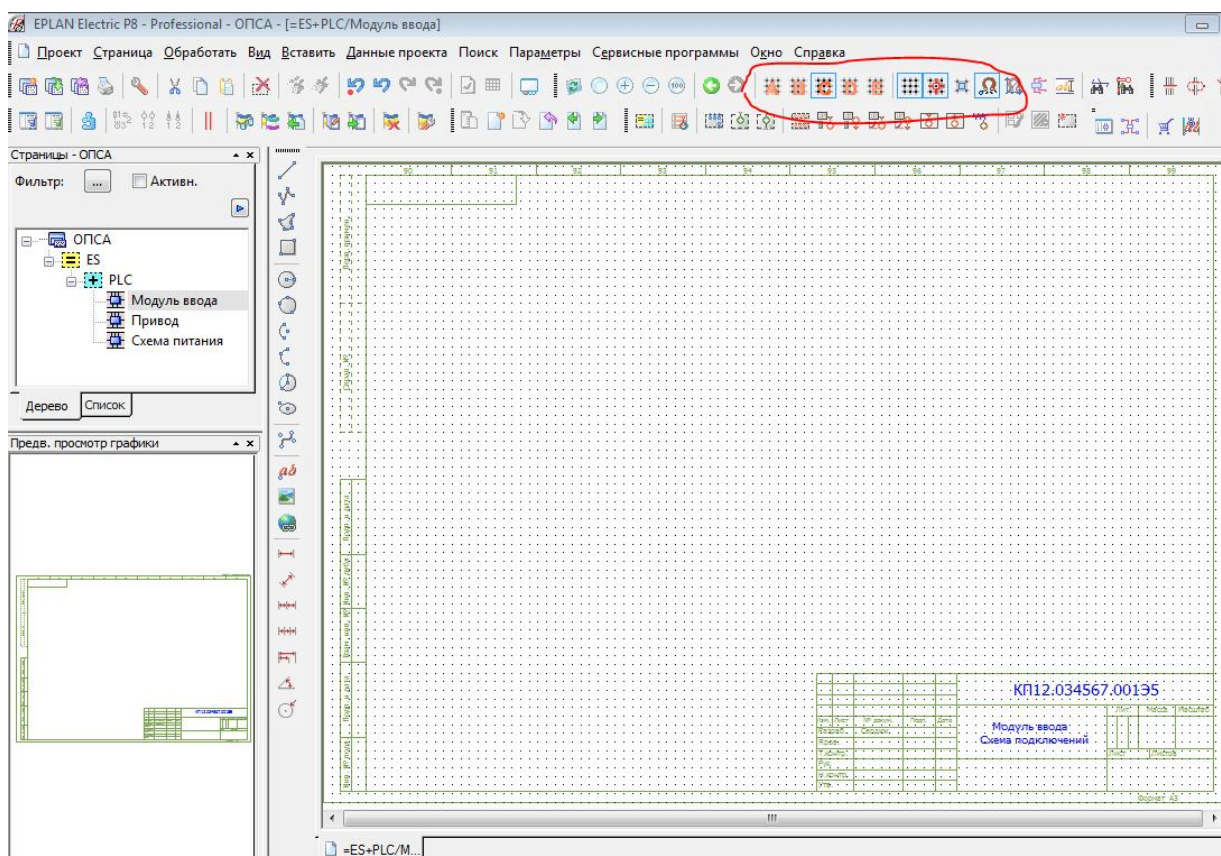


Рисунок 3.8 – Вид вікна графічного редактора при відкритій сторінці «Модуль уведення»

Для введення тексту в основний напис використовуйте команду **Вставить ► Графика ► Текст**. У діалоговому вікні, що відкрилося, призначте формат тексту й уведіть сам текст. Після натискання кнопки **ОК** набраний текст буде прикріплений до курсору для позиціонування на місці вставки. Вставка тексту відбувається по клацанню лівої кнопки миші. Після вставки слід натиснути праву кнопку миші й у контекстному меню вибрати опцію **Прервать операцию** для від'єднання тексту від курсору.

Зазвичай у вікні графічного редактора відображається ціла сторінка. При обробці схеми з'єднань найчастіше потрібно збільшити масштаб певного фрагмента екрана, щоб краще розглянути деталі зображення. Усі операції по масштабуванню можна виконувати за допомогою кнопок, показаних на рисунку 3.9.



Рисунок 3.9 – Кнопки керування масштабом відображення

Крайня зліва кнопка дозволяє збільшити фрагмент схеми, крайня справа кнопка встановлює відображення всієї сторінки, кнопки + і – здійснюють

східчасту зміну масштабу. Крім того, масштабувати зображення можна за допомогою коліщати миші.

При створенні електричних схем використовуються умовні позначки електротехнічних пристроїв і їх з'єднань.

Умовна позначка – це графічний елемент для відображення певної функції пристрою. Вона складається з функції й символу. При цьому функція містить логічні дані, а символ – графічні.

Порядок створення електричної схеми не принциповий. Однак для зручності роботи доцільно починати з побудови найбільш протяжних або найбільш великих елементів. Для побудови схеми бажано підготувати ескіз схеми, щоб заздалегідь знати щільність і порядок розміщення елементів схеми.

Слід урахувати, що проект системи автоматизації на базі програмувального логічного контролера (ПЛК) складається з десятків сторінок, на яких представляються окремі ділянки схеми. Усі ці ділянки мають зв'язки по живленню й керуванню, причому лінії зв'язку переходять із однієї сторінки на іншу з використанням *перехресних посилань*.

На кожному аркуші прийнято відображати ту частину схеми, яка має закінчену виставу в рамках прийнятої структурної ідентифікації.

Розглянемо спочатку послідовність і правила створення принципових електричних схем ПЛК.

Основними модулями ПЛК є модулі введення й виводу інформації. Ці модулі мають різне число каналів. Так, наприклад, модулі введення й виводу дискретних сигналів мають 16 і 32 каналів, модулі введення й виводу аналогових сигналів мають зазвичай 4 і 8 каналів. Враховуючи, що вистава схемотехніки великої кількості каналів модуля на одній сторінці формату А3 проблематична, на практиці застосовується спосіб роздільної вистави схем дискретного введення-виводу по 8 каналів. Із цього погляду модуль введення дискретних сигналів на 16 каналів повинен бути представлений двома сторінками.

Нумерація каналів проводиться в діапазоні від 0 до 7, тобто в межах байта. Тоді перша сторінка буде відображати підключення зовнішніх пристроїв до каналів 0...7, як частина перша (адреси входів від I 0.0 до I 0.7), а друга сторінка – підключення до каналів 0...7, як частина друга (адреси входів від I 1.0 до I 1.7).

Отже, на сторінці «Модуль уведення» слід розташувати умовне відображення модуля уведення контролера з 8 каналами, пристроїв зовнішніх підключень (датчики й вимикачі), а також лінії живлення зовнішніх пристроїв, які є загальними для схеми підключень ПЛК.

Почнемо побудову схеми з розміщення модуля контролера. Для цього виберемо в меню **Вставить ► Макрос символа ► SIE.6ES7321-1BH02-0AA0.ema**.

Обраний макрос являє собою умовну позначку модуля уведення SM 321 контролера SIMATIC S7 фірми Siemens. На рисунку 3.10 показаний фрагмент умовної позначки модуля уведення (частина 1) з адресами каналів уведення IN, нижче яких перебувають фізичні адреси пристроїв уведення (I 0.0...I 0.7).

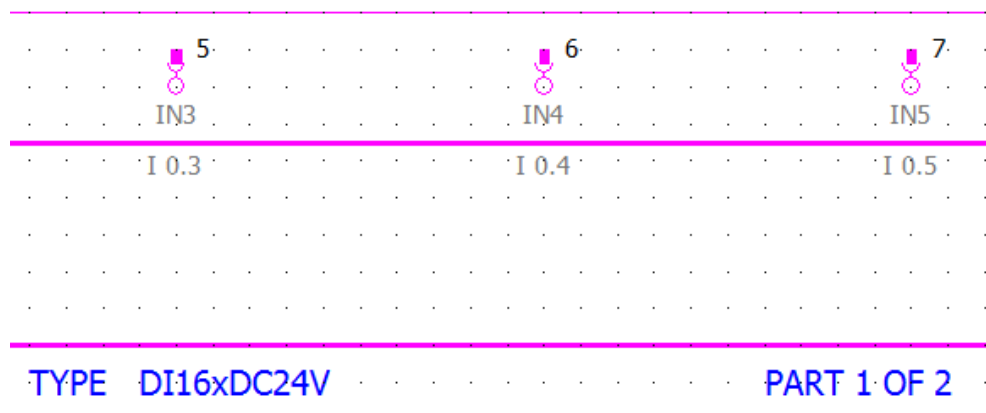


Рисунок 3.10 – Фрагмент умовної позначки модуля уведення контролера SIMATIC S7 SM 321 DI 16x24 VDC

Далі на аркуш слід нанести лінії живлення зовнішніх пристроїв (напруга 24 В). Із цією метою командою **Вставити** ► **Символ соединения** ► **Точка розрива** встановити в лівій частині аркуша (угорі) точки розриву L+ і L-, показані на рисунку 3.11.

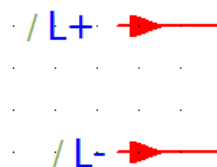


Рисунок 3.11 – Точки розриву для ліній живлення вхідних пристроїв

Похила риса перед позначенням точки призначена для поділу перехресного посилення й позначення точки розриву. У нашому випадку посилення не створене й перед похилою рисою нічого не записано.

Продовжимо створення ліній живлення. Тими ж командами вставимо точки розриву в правій частині сторінки. При вставці точок розриву слід розташовувати їх строго на тих же лініях, на яких перебувають точки лівої частини. При виконанні цієї умови лінія між точками здобуває червоне фарбування, що свідчить про автоматичний захват об'єкта. Після клацання лівої кнопки миші обидві точки автоматично з'єднуються. Натисканням правої кнопки включаємо контекстне меню й вибираємо команду **Прервать операцию**.

У результаті виконаних дій сторінка приймає вид, показаний на рис. 3.12.

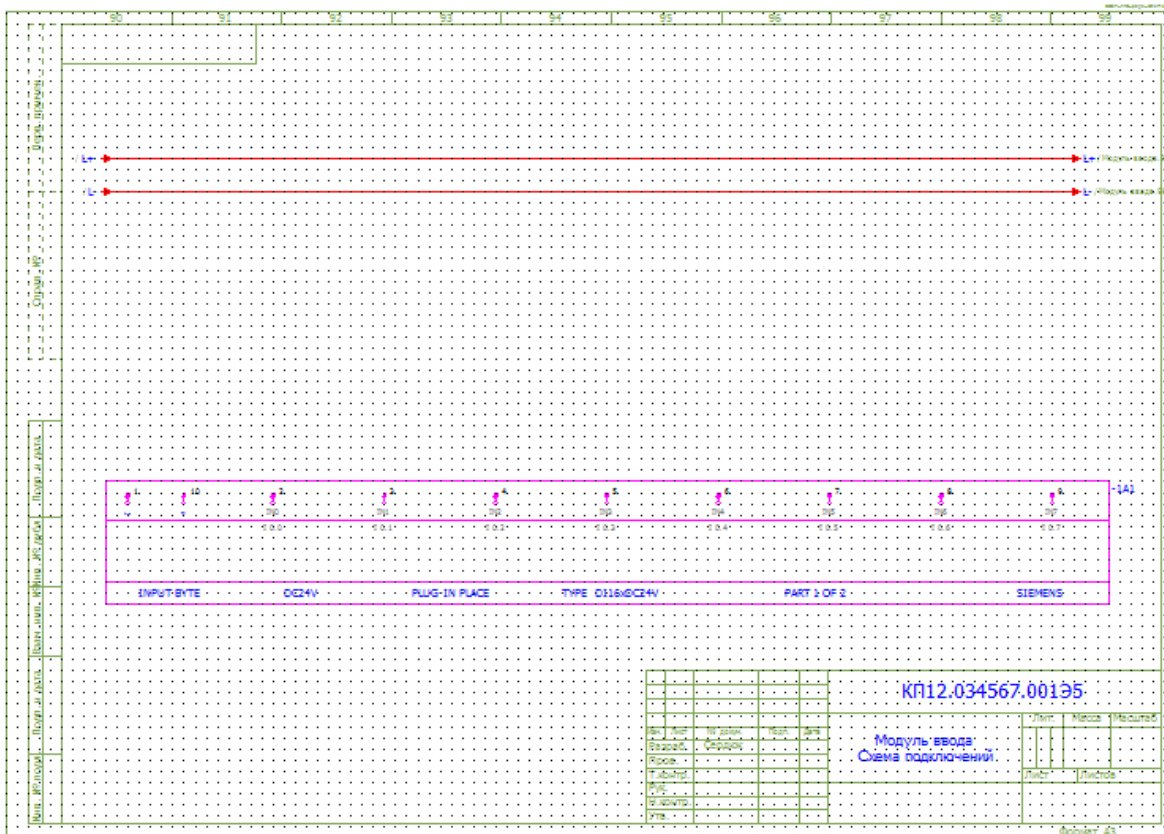


Рисунок 3.12 – Вид схеми після вставки модуля контролера й ліній підведення живлення 24 В

Наступні кроки проектування пов'язані з розміщенням пристроїв, що приєднуються до модуля контролера, і створенням з'єднань.

У редакторі EPLAN є бібліотеки макросів символів і пристроїв (виробів). У якості макросу символу може виступати окремий пристрій (функція), частина схеми й навіть уся схема. Тому макроси символів, вікон і сторінок не класифіковані й перебувають в одній папці. Для їхнього вибору використовується вікно попереднього перегляду.

Враховуючи те, що датчики є готовими виробами, скористаємося бібліотекою пристроїв. Командою **Вставити ► Устрійство** відкриємо діалогове вікно **Вибір изделия** (рис. 3.13).

Розкриємо список сигнальних пристроїв. У цьому списку виберемо відбивний оптичний перемикач із замикаючим контактом **SIEMENS 3RG7204-3CC00** і підтвердимо вибір кнопкою **ОК**. У робочій області рухаємо курсор із прикріпленням до нього датчиком так, щоб нижній вивід датчика перебував на одній вертикальній лінії з виводом IN0 модуля введення. При цьому між виводом датчика й виводом IN0 з'явиться червона лінія з'єднання. Клацаємо лівою кнопкою миші – з'єднання створене.

Повторюємо цю процедуру для всіх наступних виводів модуля (IN1...IN7). При вставці кожного нового датчика йому автоматично привласнюється наступний позиційний номер, що спрощує процес створення схеми підключення датчиків до модуля введення.

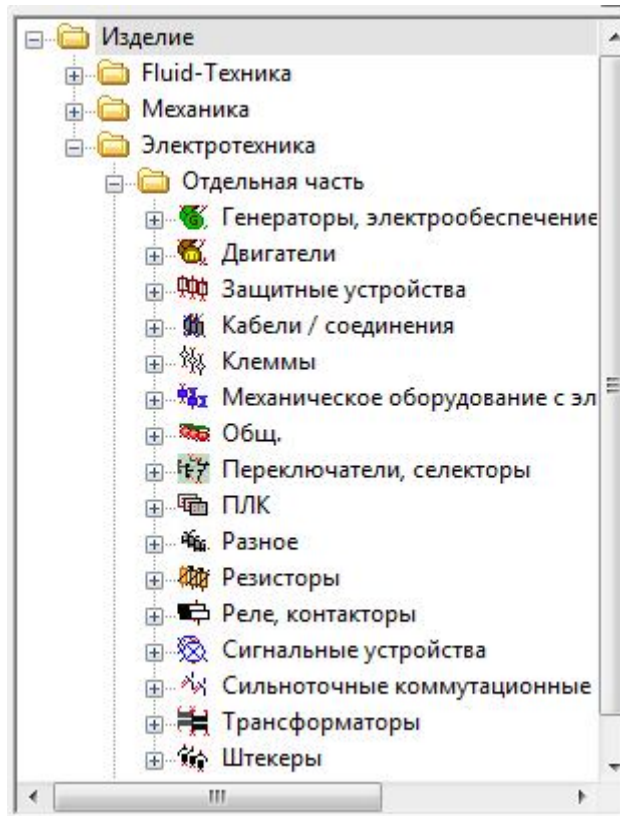


Рисунок 3.13 – Бібліотека виробів у вікні вибору виробу

Далі слід підключити всі датчики до ліній живлення L+ і L-.

Із цією метою вибираємо команду **Вставити ► Символ соединения ► Тройник внизу**, а простіше – натискаємо **F7**. У результаті виводиться вікно **Тройник внизу** (рис. 3.14), у якому можна вибрати варіант «Начертить как точку» або варіант «1-ая цель слева, 2-ая цель справа».

Вибравши другий варіант, розташовуємо курсор на лінії L+ проти виводу 1 датчика S1. З появою червоної лінії між цими виводами клацаємо лівою кнопкою, фіксуючи з'єднання. Далі переводимо курсор на лінію L- напроти виводу 2 датчика й створюємо друге з'єднання. Повторюємо процедуру для всіх датчиків і одержуємо схему, фрагмент якої показано на рисунку 3.14.

Завершуємо створення схеми підключень модуля введення промальовуванням таблички для опису призначення датчиків, що підключаються.

Готова схема представлено на рисунку 3.15.

3.4 Створення схем з'єднань

Схема з'єднань (монтажна) – це схема, що показує з'єднання складових частин виробу (установки) і визначає проведення, джгути та кабелі, якими

здійснюються ці з'єднання, а також місця їх приєднання й уведення (затискачі, рознімання, сальники, прохідні ізолятори й т.п.).

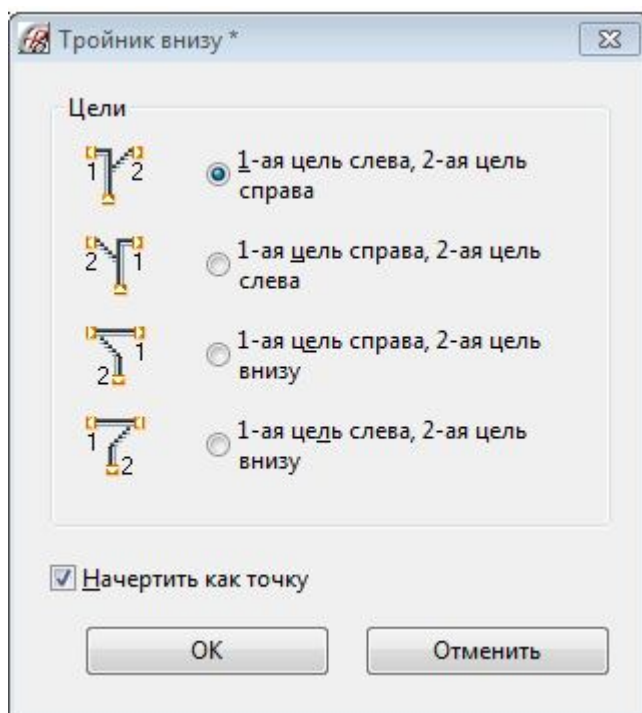


Рисунок 3.14 – Вид вікна *Тройник внизу*

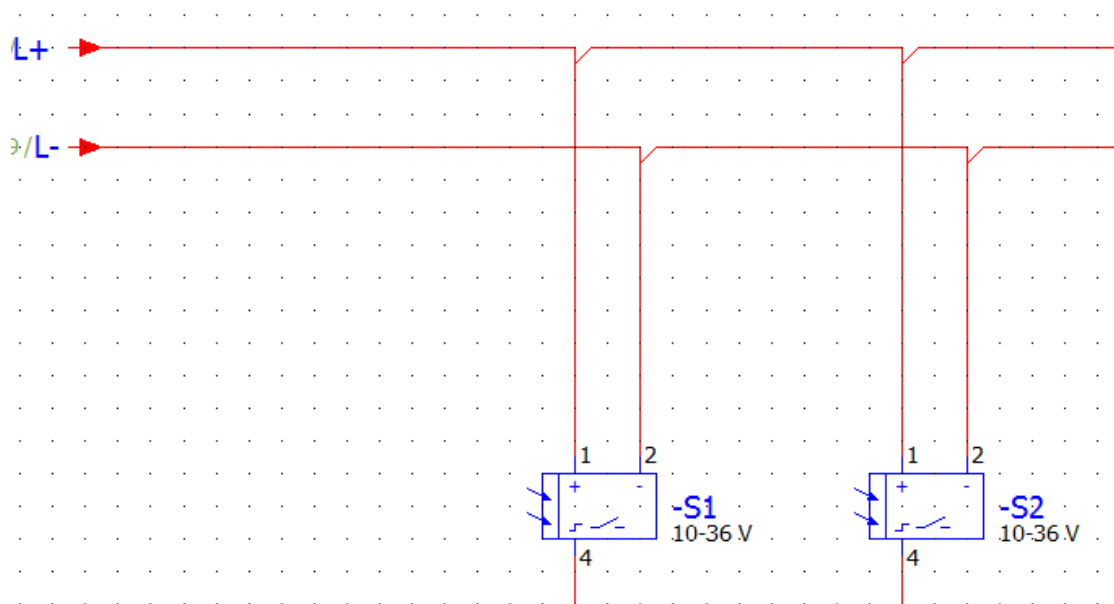


Рисунок 3.15 – Фрагмент схеми підключень датчиків

Схема з'єднань є основним документом, передбаченим ГОСТ, по якому монтажники роблять складання електричних шаф і панелей, а також з'єднання електроустановок.

Слід прийняти до уваги, що системи проектування в області електротехніки, які, в основному, створюються західними фірмами, не

передбачають розробку монтажних схем – західним фірмам вони просто не потрібні. Усе, що необхідно для монтажу, показується на схемі електричній принциповій, яка виконується по стандарту Міжнародної електротехнічної комісії IEC (International Electrotechnical Commission).

Відмінність принципової схеми за ГОСТ і принципової схеми по стандарту IEC полягає в різній виставі символів з'єднань.

За ГОСТ на принциповій схемі символ з'єднання відображається як *точка* з'єднання. На рисунку 3.16,а показаний приклад з'єднання двох апаратів за ГОСТ. Із цього прикладу видно, що з'єднання існує, однак не можна сказати, як проходять проведення від апарата до апарата.

По міжнародному стандарту IEC те ж саме з'єднання відображається інакше (рис. 3.16,б).

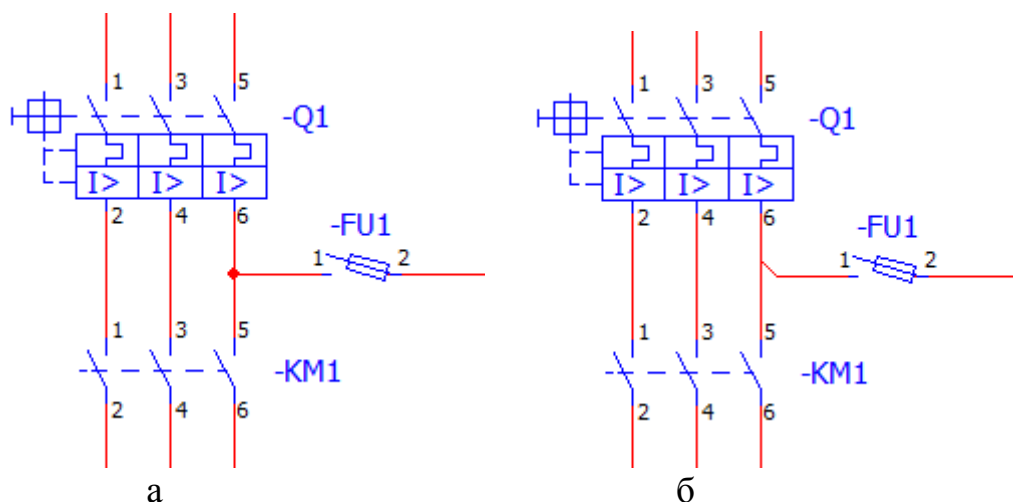


Рисунок 3.16 – Виконання з'єднань на принципових схемах

Тут питання про проходження провідника від апарата до апарата не виникає, тому що зі схеми все ясно й зрозуміло. На жаль, відповідно до нормативних документів ГОСТ, схема принципова із застосуванням цього символу не пройде заслін нормоконтролю. Тому для виробництва монтажних робіт замовники вимагають схему з'єднань.

У процесі створення схеми з'єднань найбільша увага проектувальника приділяється роботі із *символами з'єднань*, тому що апаратура й усі складові частини електроустаткування вже визначені при створенні принципових схем і схем підключень.

Суть роботи із символами з'єднань полягає в тому, що на схемі повинні бути відображені не тільки лінії з'єднань, але й зазначені всі необхідні їхні параметри – *потенціали, марки проводів і кабелів, перетини проводів, колір і нумерація, а також зазначені клеми й штекери.*

Призначення потенціалів.

Призначення потенціалів корисно для їхнього відстеження на схемах і для автоматичної нумерації з'єднань. За допомогою визначення потенціалів можна швидко простежити шляхи поширення сигналів на схемі завдяки тому, що при подачі команди **Отслеживание потенциала** всі з'єднання з однаковим потенціалом відображаються на схемі однаковим кольором. Крім того, при автоматичній нумерації з'єднань можна привласнити однакові номери з'єднанням з однаковими потенціалами.

Визначення потенціалу проводиться або точкою визначення потенціалу, або шляхом вставки потенціалу.

Розглянемо приклад створення виводів потенціалів трифазної мережі L1, L2, L3 з нульовим проведенням N (нейтраллю) і заземленням PE (рис. 3.17).

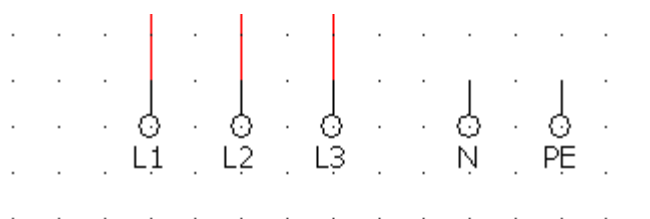


Рисунок 3.17 – Приклад створення виводів потенціалів

Вставка приєднання потенціалу в схему з'єднань проводиться в такий спосіб.

1. Виберіть пункт меню **Вставити > Вывод потенциала**. Символ приєднання потенціалу з'явиться поруч із курсором миші.

2. Поставте курсор на місце першого приєднання й клацніть по ньому лівою кнопкою миші. Відкриється діалогове вікно **Свойства (усл. обозначения): Подсоединение потенциала**, показане на рисунку 3.18.

3. Уведіть на вкладці **Определение потенциала** в поле **Имя потенциала** перше ім'я – L1, а в груповому полі **Свойства** зі списку **Значение**, що розкривається, виберіть значення – L.

4. Перейдіть на вкладку **Данные символа/функция** (рис. 3.19).

5. У полі **Номер/имя** клацніть по кнопці [...]. Відкриється діалогове вікно **Выбор символа** (рис. 3.20), у якому виберіть відображуваний символ приєднання потенціалу. Клацніть по кнопці **ОК** для закриття вікна **Выбор символа** й продовжуйте роботу на вкладці **Данные символа/функция** діалогового вікна **Свойства**. Розкрийте список **Вариант**: і виберіть бажаний напрямок виводу приєднуемого потенціала (A, B, C, D), який відображується у полі попереднього перегляду.

6. Перейдіть на вкладку **Отображение**. Тут відкрийте список **Порядок свойств** (рис. 3.21) і виберіть місце для вказівки властивостей потенціалу (тексту) щодо його графічного відображення.

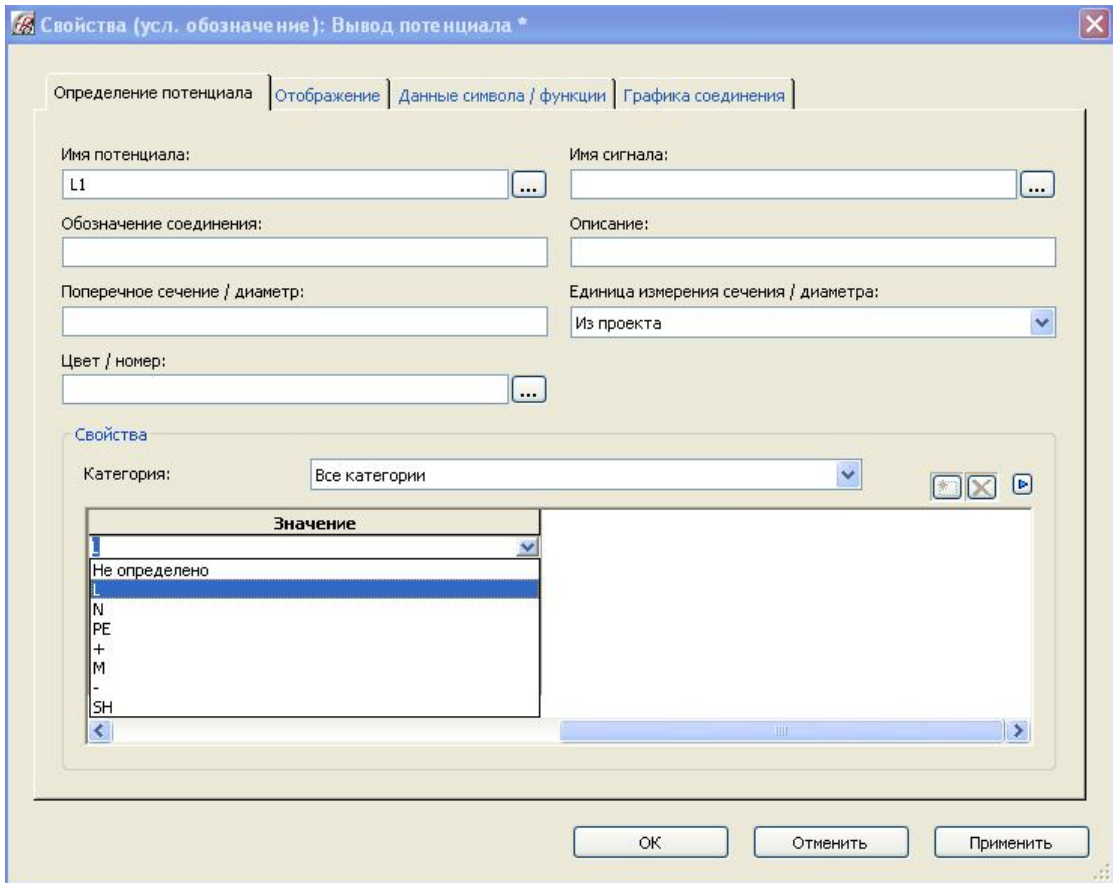


Рисунок 3.18 – Вид вікна для позначення виводу потенціалу

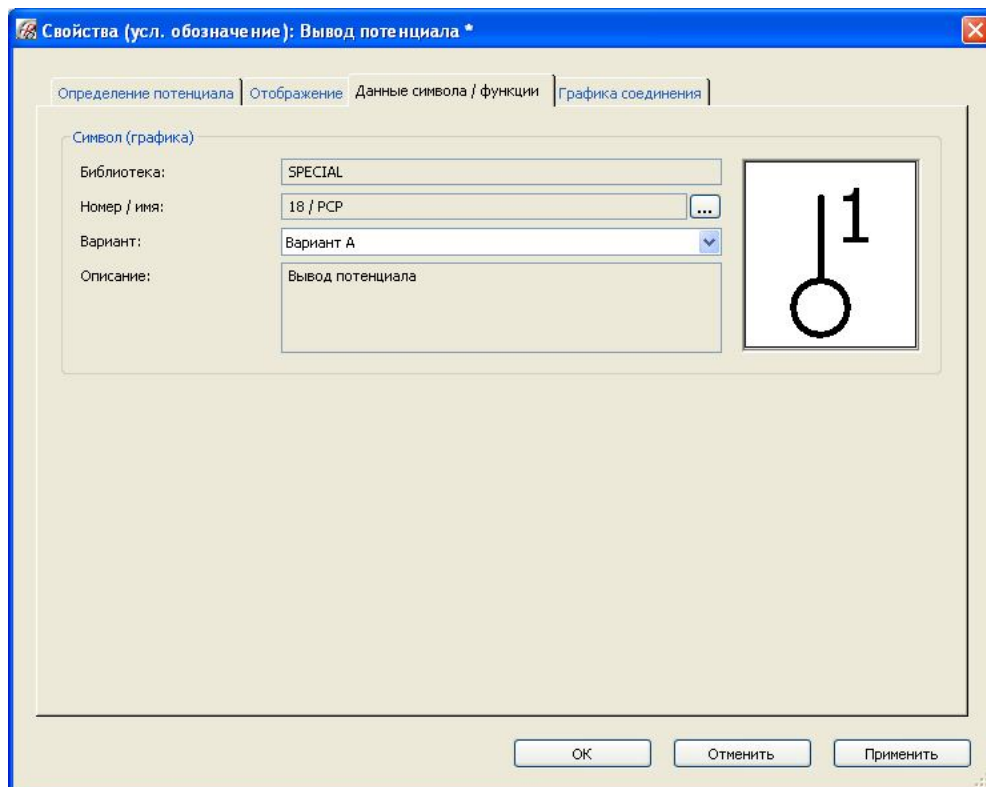


Рисунок 3.19 – Вид вкладки *Данные символа/функции*

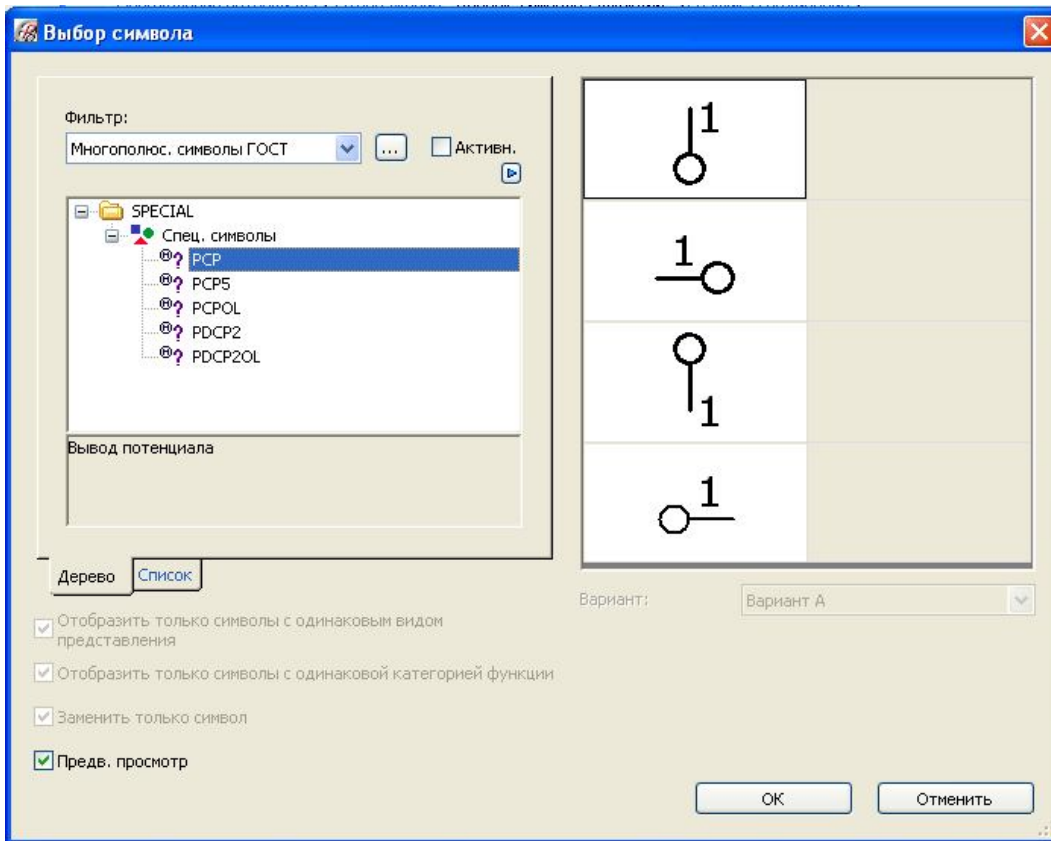


Рисунок 3.20 – Вид вікна **Выбор символа**

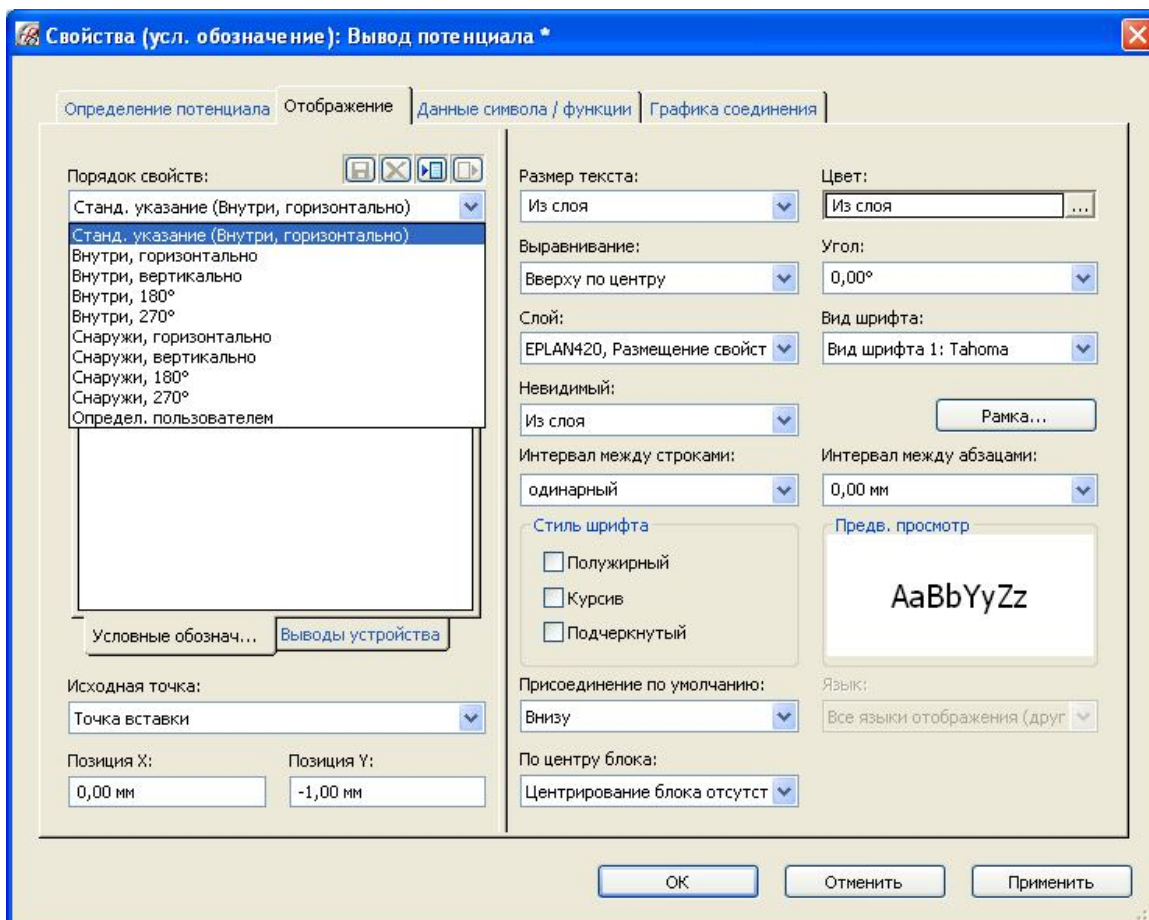


Рисунок 3.21 – Вид вікна **Свойства...** на вкладці **Отображение**

7. Далі задайте перелік властивостей – він відображається на полі вкладки **Условные обознач...** у вигляді списку (**Имя потенциала**, **Имя сигнала** і т.д.). Для видалення зайвих властивостей клацніть на полі правою кнопкою й у контекстному меню виберіть **Удалить**, а для додавання властивостей виберіть **Добавить**, як показано на рисунку 3.22.

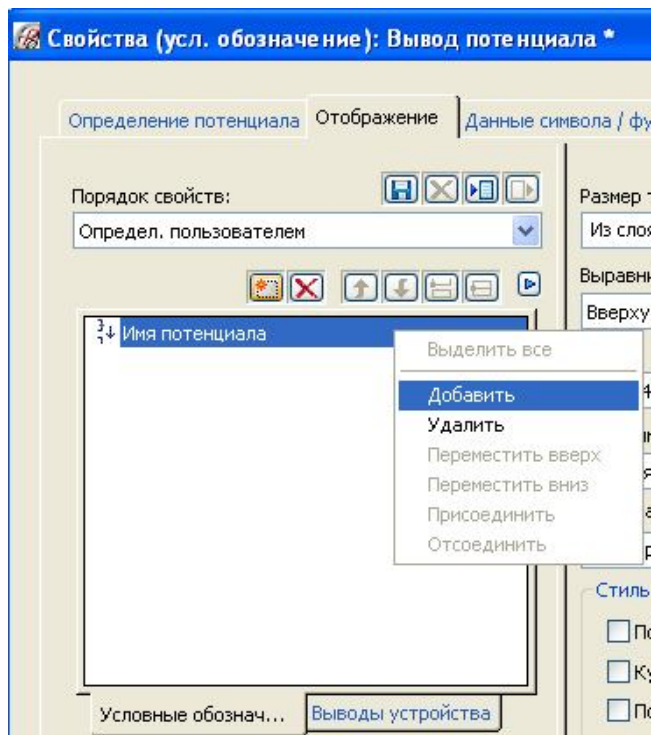


Рисунок 3.22 – Вибір операції Додати

8. По команді **Добавить** відкриється вікно **Выбор свойств**, показане на рисунку 3.23. У цьому вікні можна вибрати ті властивості, які повинні бути відображені поруч із символом потенціалу. Завершіть цю операцію, натиснувши кнопку ОК.

9. Після призначення властивостей потенціалу L1 потрібно вставити наступні приєднання – L2, L3 зі значеннями L, а також приєднання N і PE зі значеннями N і PE, відповідно.

Потенціал можна задати також *точкою визначення потенціалу*. Ця точка може бути видимою (похила риска на лінії з'єднання) і невидимою. Після вставки точки визначення потенціалу (команда **Вставить ► Точка определения потенциала**) відкривається вікно **Свойства (усл. обозначения): Точка определения потенциала**. Це вікно практично ідентичне вікну **Свойства (усл. обозначения): Присоединение потенциала**, робота з яким розглянута вище. Видимість точки настраюється на вкладці **Данные символа/функции**.

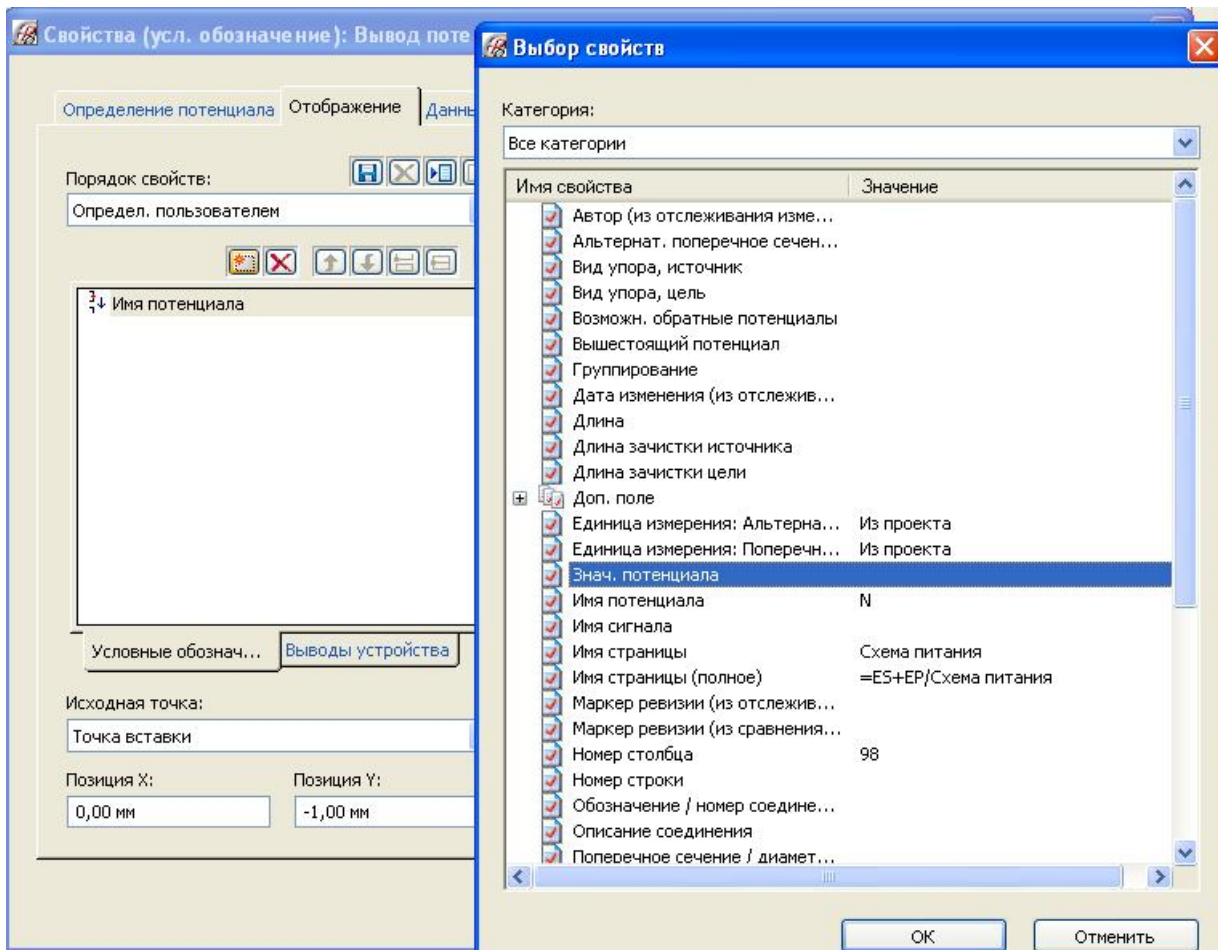


Рисунок 3.23 – Вид вікна **Выбор свойств** виводу потенціалу

Призначення властивостей кабелів.

На рисунку 3.24 показаний приклад визначення кабелю W1. Кабель складається з 5 жил перетином по 4 мм². Його електрична міцність становить 750 В, довжина – 5м.

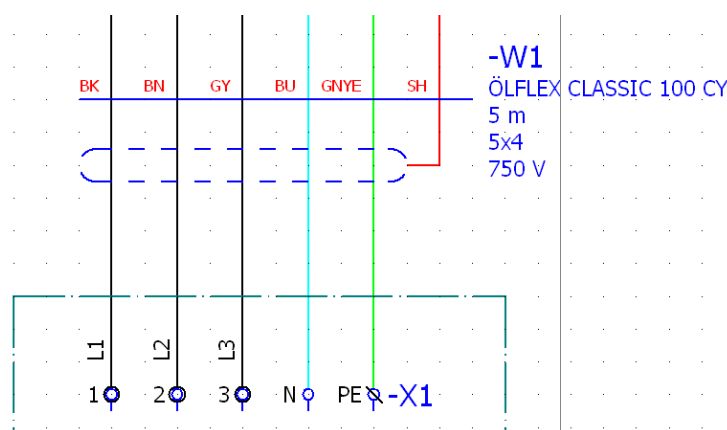


Рисунок 3.24 – Приклад визначення кабелю

Щоб призначити ці властивості, необхідно застосувати команду **Вставить ► Определение кабеля**. При цьому до курсору прикріплюється символ визначення кабелю. Далі потрібно розташувати курсор ліворуч від

ліній, які повинні ввійти в кабель, клацнути ліву кнопку миші для фіксації початкової точки, протягнути лінію через усі з'єднання й клацнути ліву кнопку в кінцевій точці. Вікно **Свойства (усл. обозначения): Кабели**, показане на рисунку 3.25, відкриється автоматично.

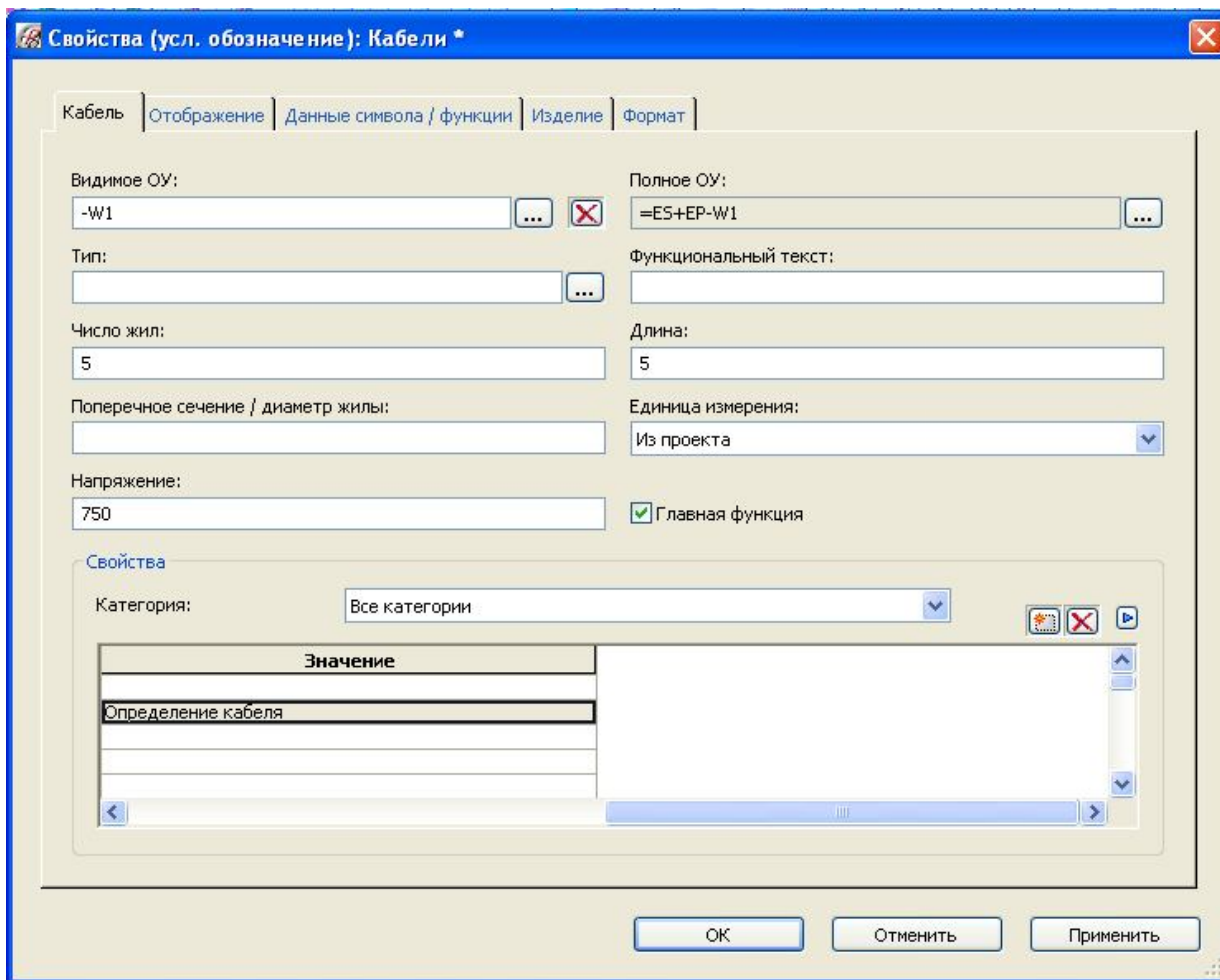


Рисунок 3.25 – Вид вікна **Свойства: Кабели** на вкладці **Кабель**

На першій вкладці **Кабель** слід призначити позначення кабелю (W1) і його технічні характеристики. Далі перейти на вкладку **Изделие** (рис. 3.26) і натиснути кнопку **Выбор устройства**. При цьому відкриється вікно **Выбор устройства...**, показане на рисунку 3.27.

У вікні **Выбор устройства** потрібно встановити мітку **Активн.** для автоматичного фільтра. При цьому в поле **Изделие** будуть відображені вироби, що відповідають заданим технічним характеристикам (у нашому випадку на рисунку 3.27 редактором відібрано один виріб). У нижній частині вікна приводяться характеристики цього виробу – кількість жил, їх колір, поперечний перетин, тип потенціалу й інше.

Після підтвердження вибору виробу редактор закриває вікно **Выбор устройства**, а на вкладці **Изделие** відкритого вікна **Свойства: Кабели** з'являються дані обраного кабелю.

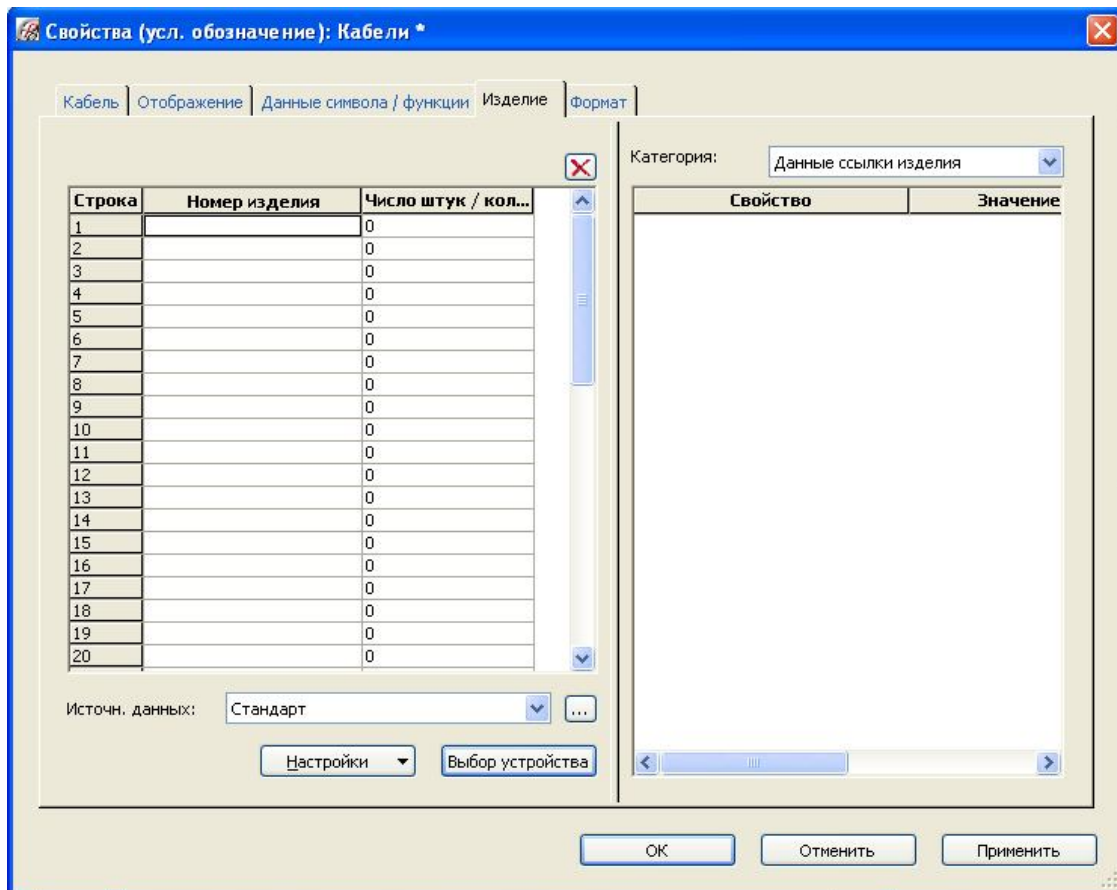


Рисунок 3.26 – Вид вікна **Свойства: Кабели** на вкладці **Изделие**

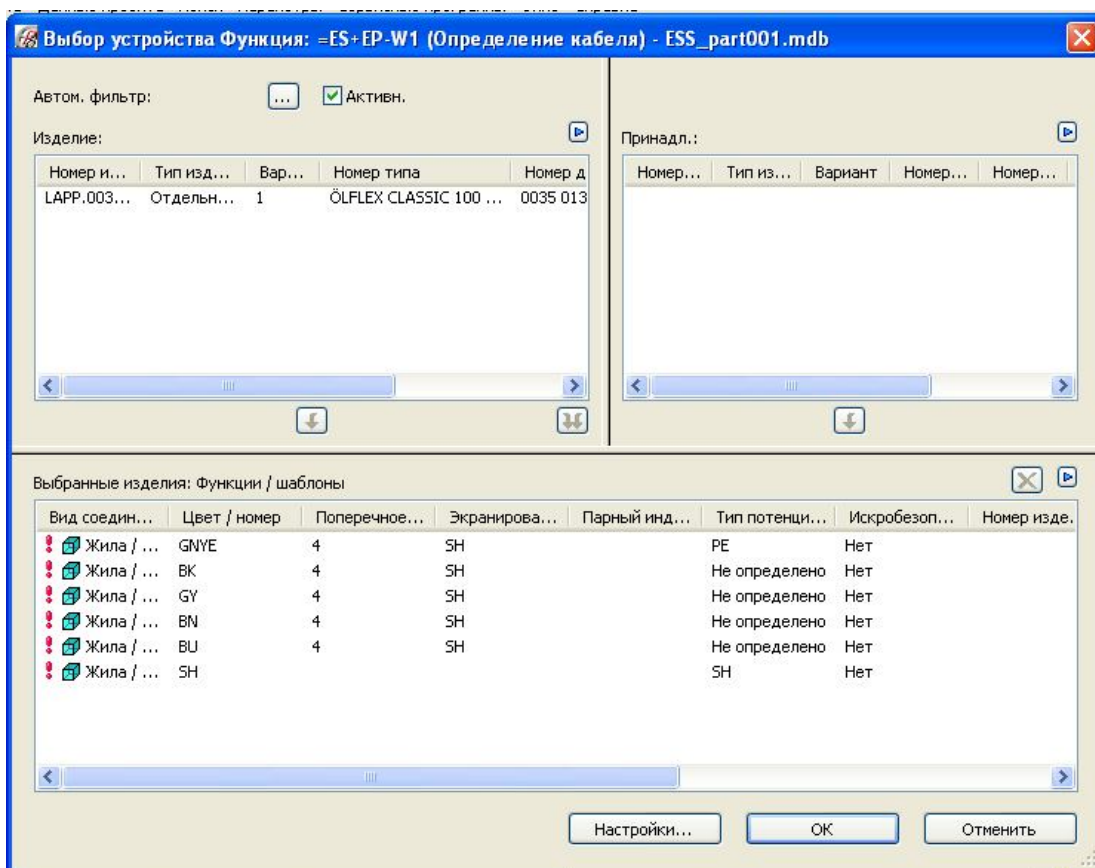


Рисунок 3.27 – Вид вікна **Выбор устройства** з даними кабелю

Наступним кроком у процесі призначення кабелю є настроювання відображення лінії визначення кабелю. Для цього переходимо на вкладку **Отображение**, вид якої показано на рисунку 3.28. Тут слід відредагувати властивості, які повинні бути відображені біля лінії визначення кабелю.

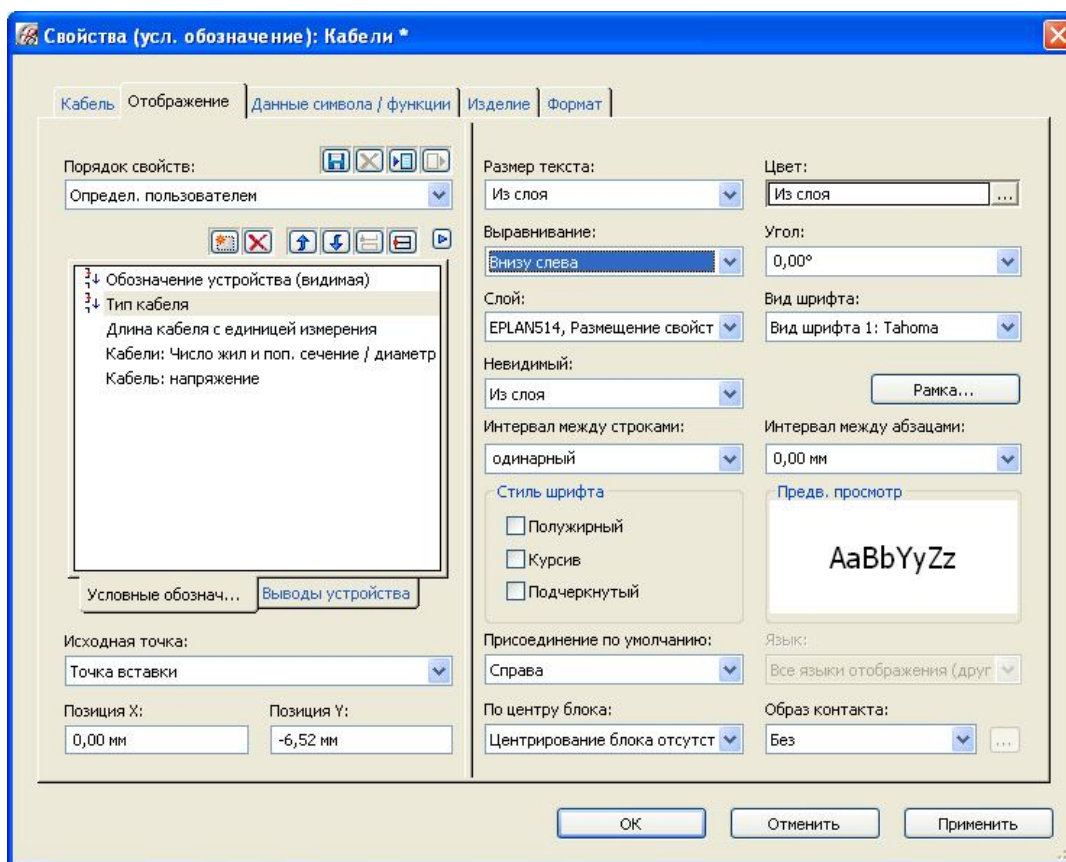


Рисунок 3.28 – Вид вкладки **Отображение** у вікні **Свойства: Кабели**

Суть редагування полягає у встановленні переліку властивостей і порядку їх розміщення на схемі. Для цього в поле **Условные обознач.** за допомогою контекстного меню необхідно додати потрібні або вилучити непотрібні властивості, потім відкрити список **Порядок свойств** і вибрати місце їх розташування на схемі щодо лінії визначення кабелю.

Далі слід перейти на вкладку **Данные символа/функции**, яка показано на рисунку 3.29.

На цій вкладці задається графіка лінії визначення кабелю – лінія (як на рисунку) або коло. Вибір графіки проводиться у вікні **Выбор символа**, яке відкривається кнопкою [...] у рядку **Номер/имя**.

На цьому визначення кабелю й настроювання його властивостей завершуються.

Призначення властивостей проводів.

Для призначення властивостей проводам, що створюють з'єднання між пристроями, використовуються *точки визначення з'єднань*. За допомогою цих точок можна задати поперечний переріз або діаметр провідника, його довжину,

колір ізоляції, а також позначити номер з'єднання. Усі ці дані використовуються для виконання монтажних робіт.

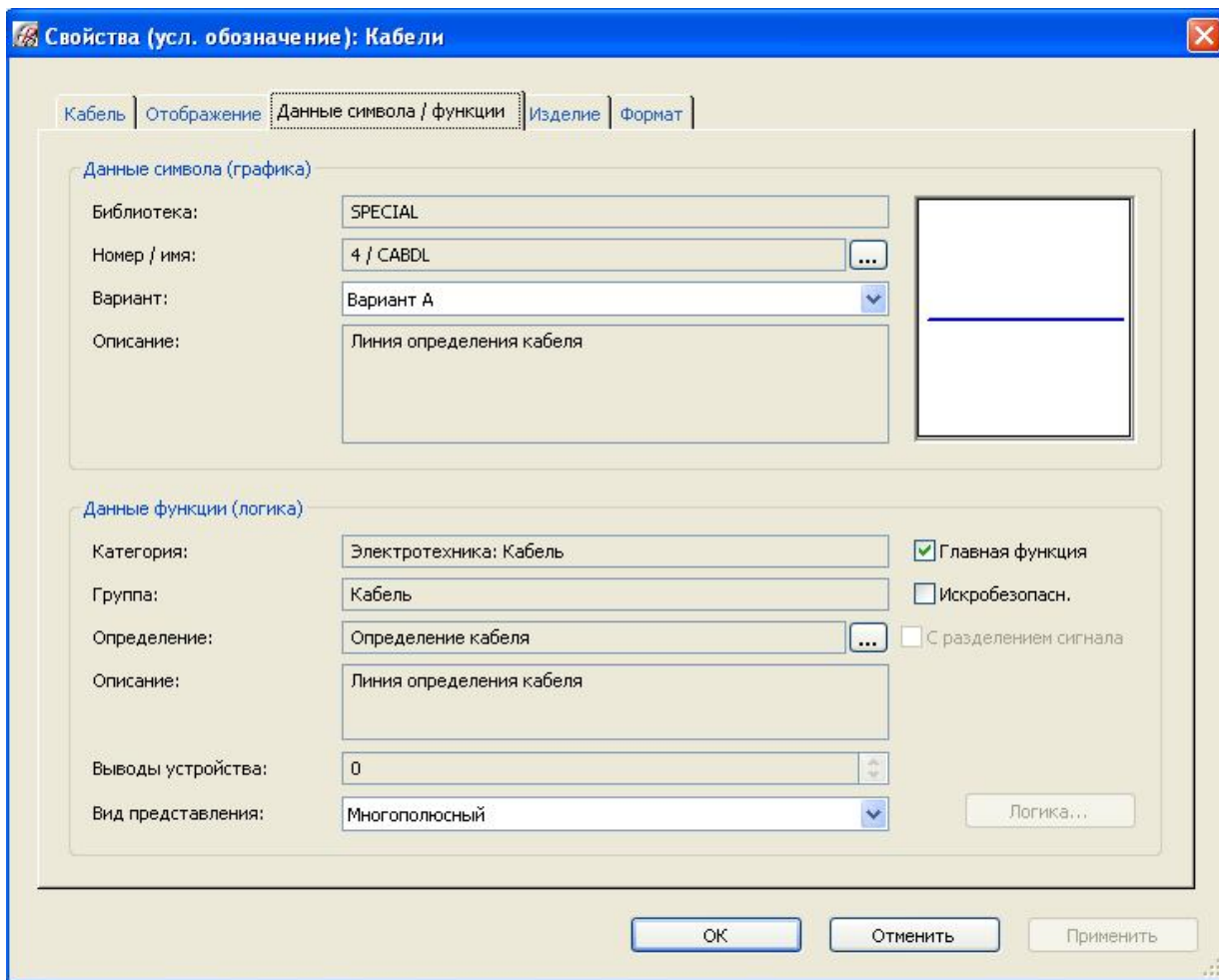




Рисунок 3.29 – Вид вкладки *Данные символа/функции*

Для створення точок з'єднань використовуйте команду **Вставить** ► **Точка определения соединения**. По цій команді відкривається вікно **Свойства...**, показане на рисунку 3.30.

На вкладці **Точка определения соединения** відзначені поля, у яких можна задати номер з'єднання, колір і поперечний переріз провідника.

На вкладці **Отображение** (рис. 3.31) у поле **Порядок свойств** можна встановити перелік і порядок властивостей, які будуть відображені на схемі. Якщо деякі властивості не потрібно відображати на схемі, то в поле **Невидимый** слід поставити оцінку **Да**, як показано на рисунку для властивості **Цвет соединения**.

Для зміни порядку проходження властивостей у списку використовуйте кнопки  , показані на рисунку 3.32.

На вкладці *Данные символа/функции* потрібно відкрити вікно вибору символу відображення точки з'єднання. Для цього натисніть кнопку [...] у полі **Номер/имя** (рис. 3.33).

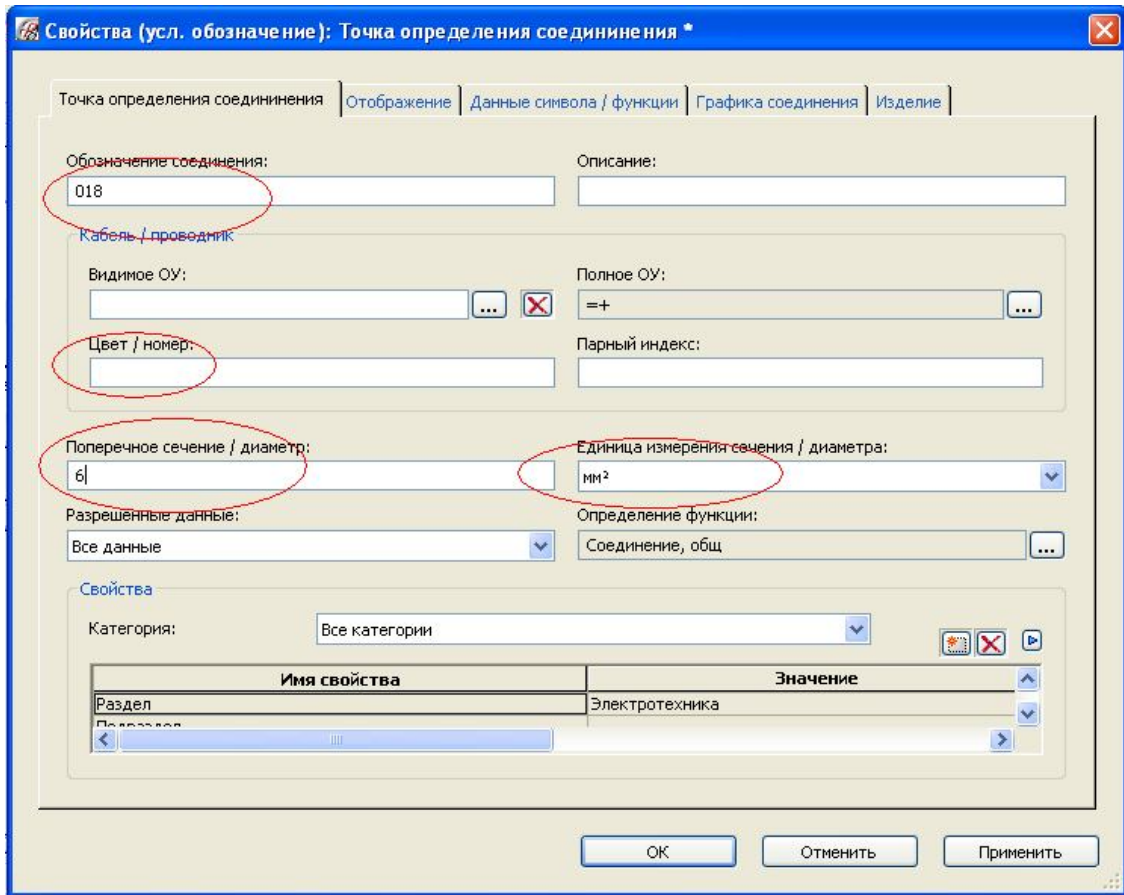


Рисунок 3.30 – Вид вкна *Свойства: Точка определения соединения*

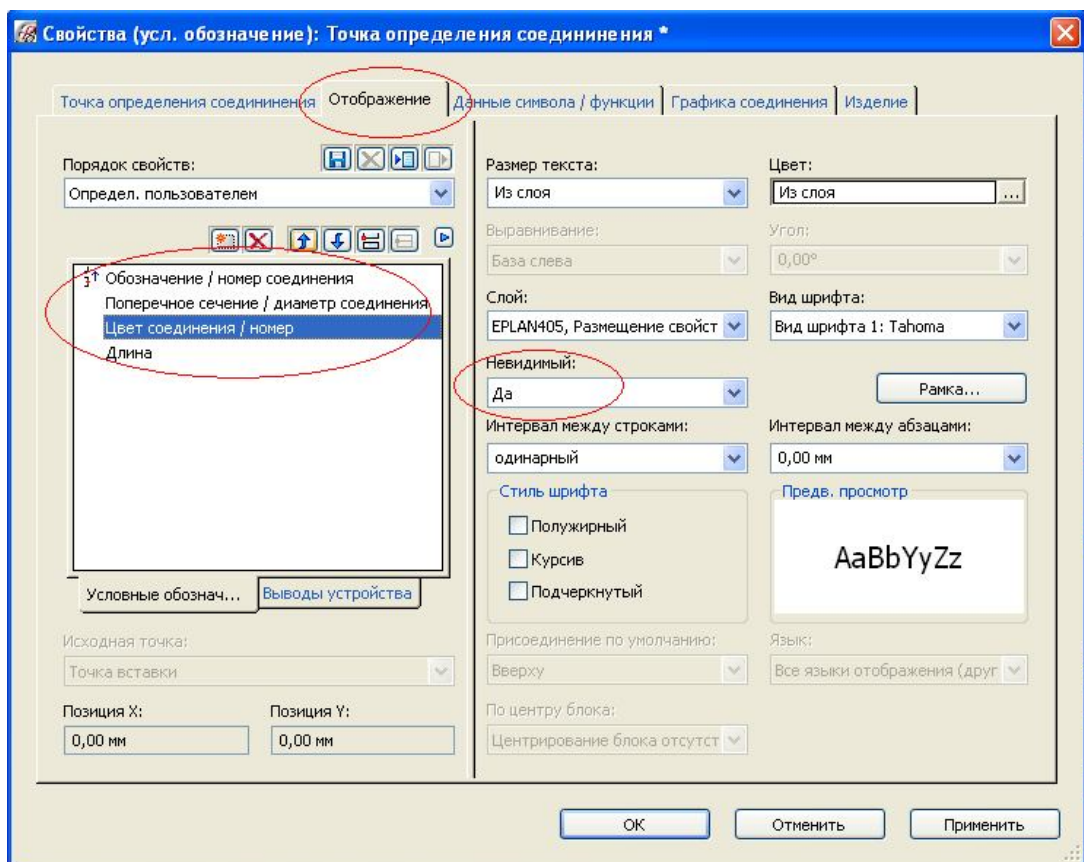


Рисунок 3.31 – Вид вкладки *Отображение* вкна *Свойства*

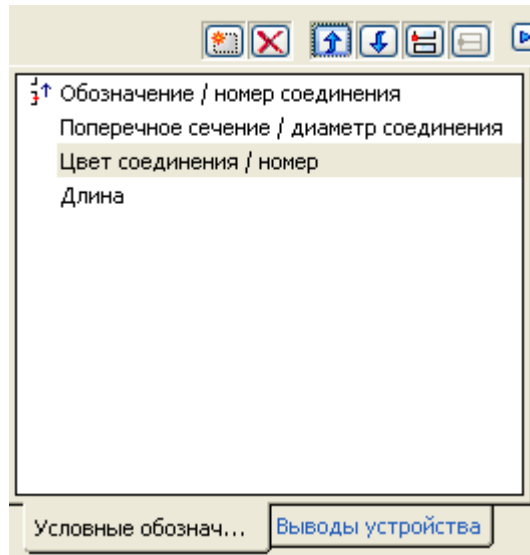


Рисунок 3.32 – Поле **Порядок свойств** із кнопками керування

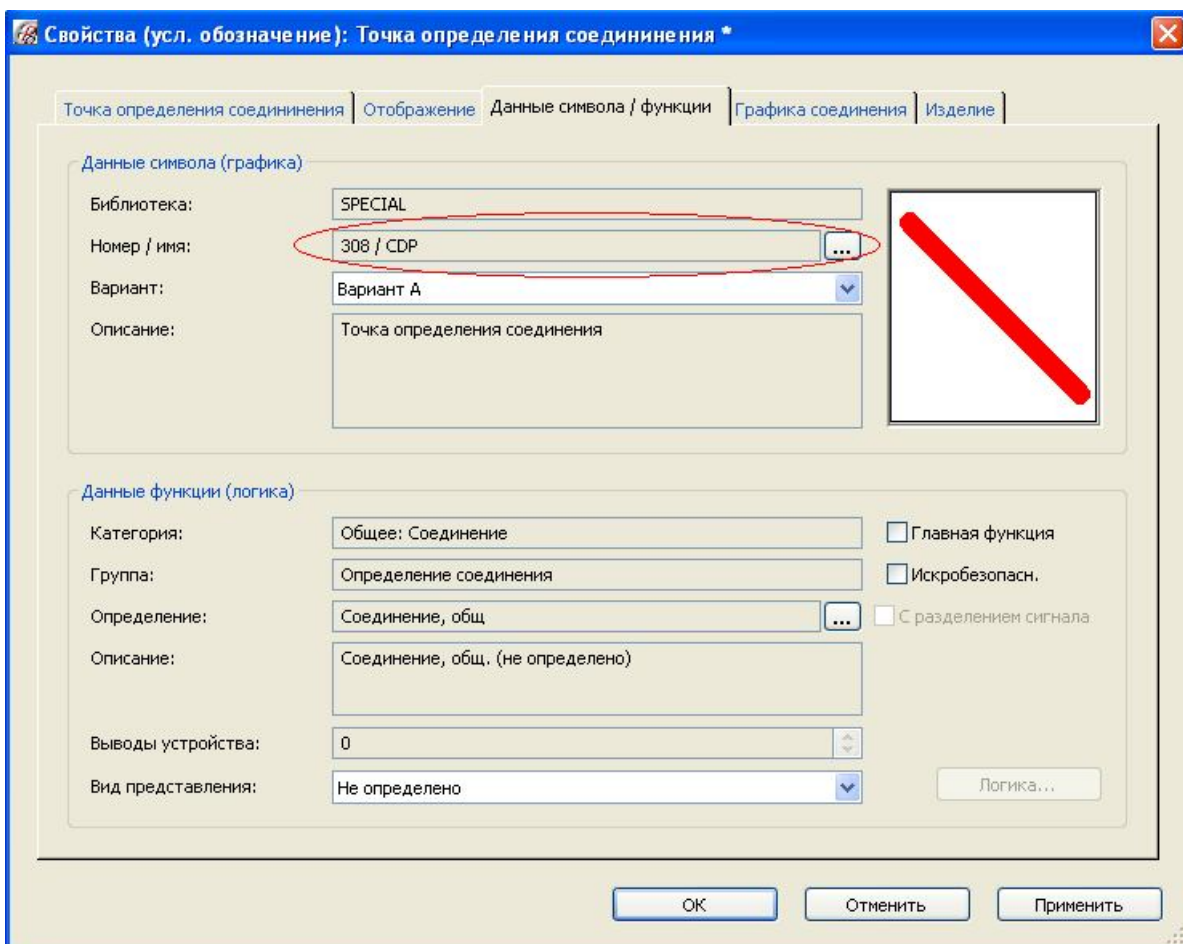


Рисунок 3.33 – Вибір графіки символу на вкладці **Данные символа**

При натисканні кнопки відкриється вікно **Выбор символа** (рис. 3.34), у якому потрібно вибрати один з варіантів вистави точки визначення з'єднання, наприклад, CDP.

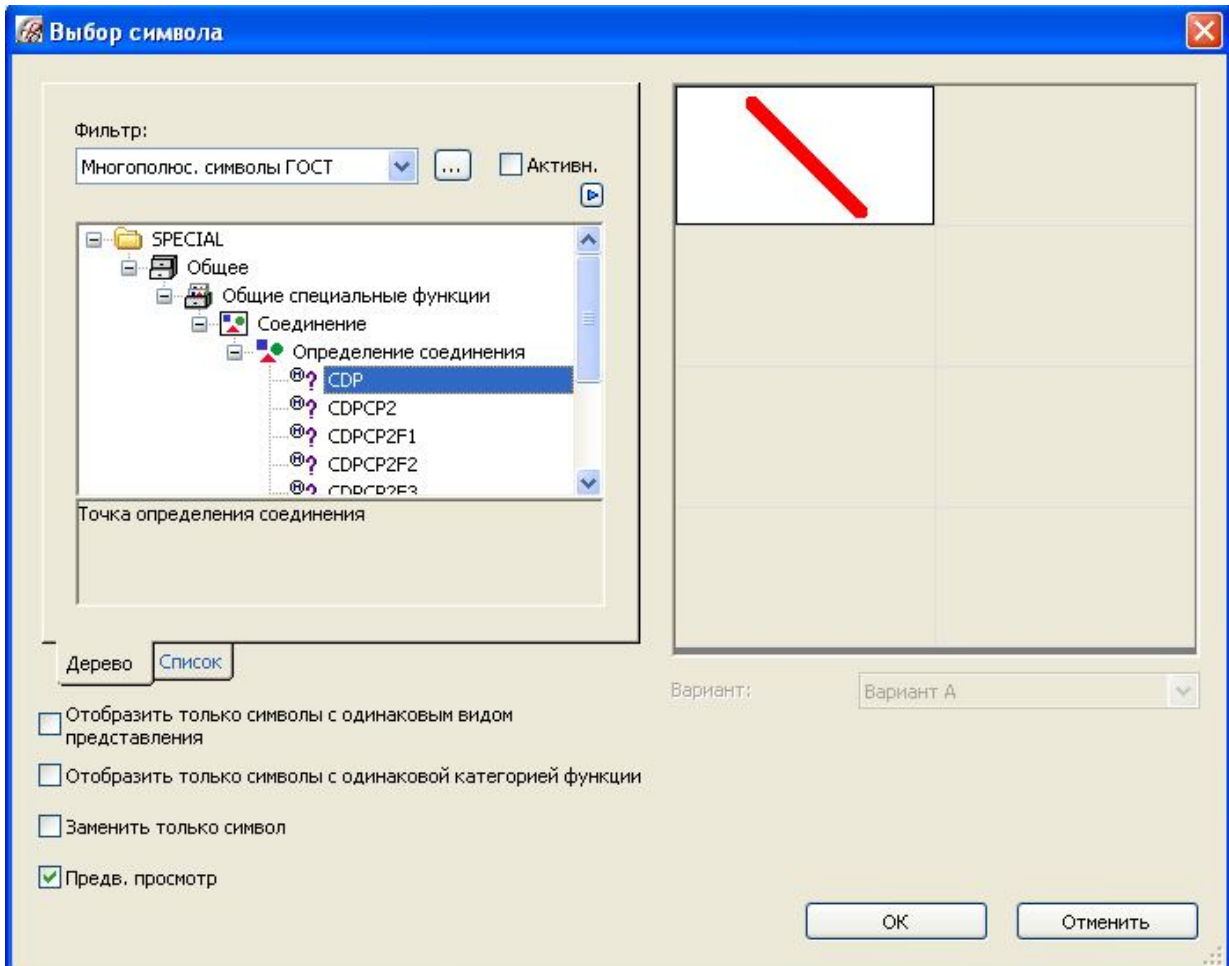


Рисунок 3.34 – Вибір графіки у вікні **Выбор символа**

Слід урахувати, що EPLAN передбачає налаштування, які будуть діяти в рамках усього проекту. За допомогою цих налаштувань можна задати властивості й відображення практично всіх елементів проекту, у тому числі й з'єднань. Так, наприклад, для налаштування нумерації з'єднань необхідно застосувати команду **Параметры ► Настройки ► Проект ► «Имя проекта» ► Соединения ► Нумерация соединений**. У вікні, що відкрилося (рис. 3.35) можна задати:

- ✓ *символ* для точки визначення з'єднання на вкладці **Размещение**;
- ✓ *схему позначення з'єднань* (з орієнтацією на потенціал, на сигнал, на сторінку або стандартну) на вкладці **Обозначение**;
- ✓ *параметри відображення з'єднання* на вкладці **Отображение**.

На цьому визначення кабелю й налаштування його властивостей завершуються.

Автоматична нумерація з'єднань.

EPLAN дозволяє також зробити *автоматичну нумерацію* всіх з'єднань. Ця операція проводиться в такий спосіб.

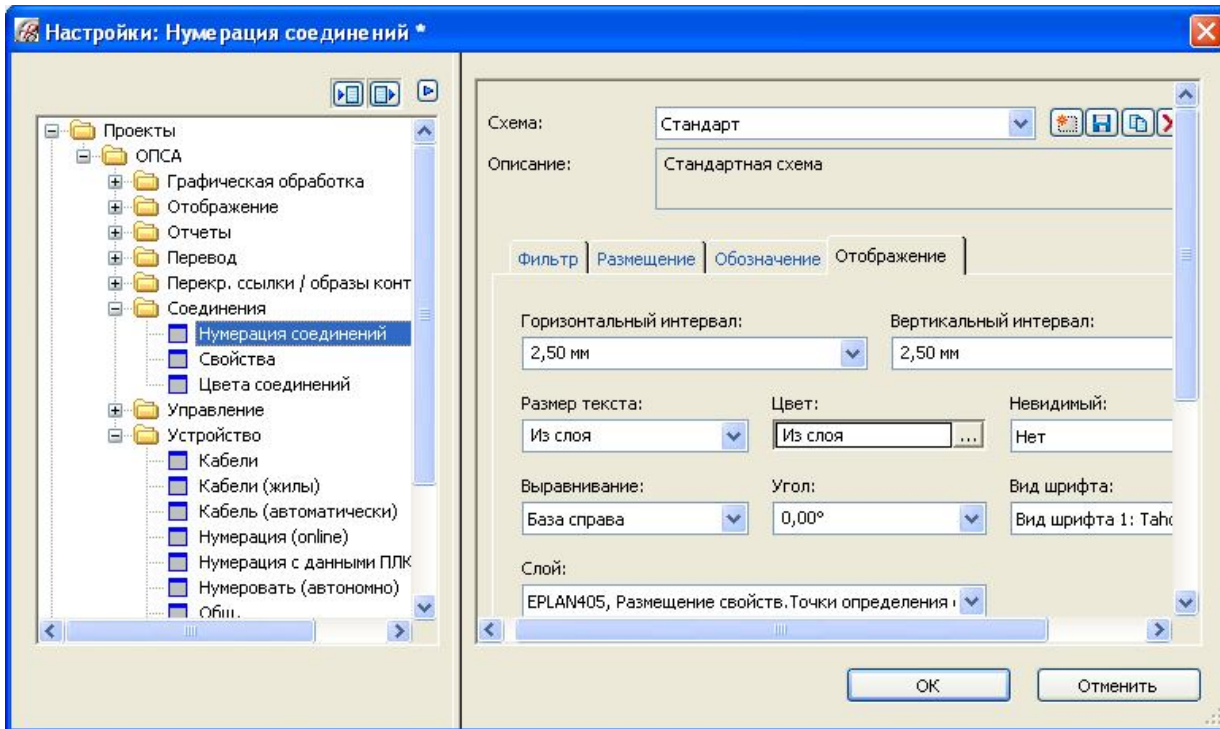


Рисунок 3.35 – Вид вікна *Настройки: Нумерация соединений*

1. Виділіть сторінку, на якій потрібно пронумерувати з'єднання, Для цього застосуйте команду **Обработать ► Выделить ► Страница**. При цьому по контуру сторінки створюється темна рамка.

2. Застосуйте команду **Данные проекта ► Соединения ► Нумерация ► Разместить**. По цій команді будуть розставлені символи з'єднань із питаннями замість номерів. Переконайтеся, що символи з'єднань розставлені правильно.

3. Застосуйте команду **Данные проекта ► Соединения ► Нумерация ► Обозначить**. У результаті питання біля символів з'єднань будуть замінені номерами, починаючи з 001. Одночасно буде створений список усіх з'єднань.

На рисунку 3.36 показаний фрагмент схеми з нумерацією з'єднань.

Размещение клем.

Клеми ставляться до пристроїв і можуть вставлятися в схему з'єднань, як устаткування загального призначення.

На схемі з'єднань зазвичай показується апаратура керування й устаткування, з яким вона зв'язана. Апаратура керування (ПЛК, захисні пристрої, реле, контактори, трансформатори, перетворювачі й т.п.) монтується в шафі керування, а встаткування перебуває за межами шафи. Таким чином, шафа повинна бути виконана, як закінчений виріб системи зі входами і виходами.

На вході повинні бути лінії з'єднання із пристроями введення (датчики, кнопки керування й т.п.), а на виході – лінії з'єднання з виконавчими пристроями.

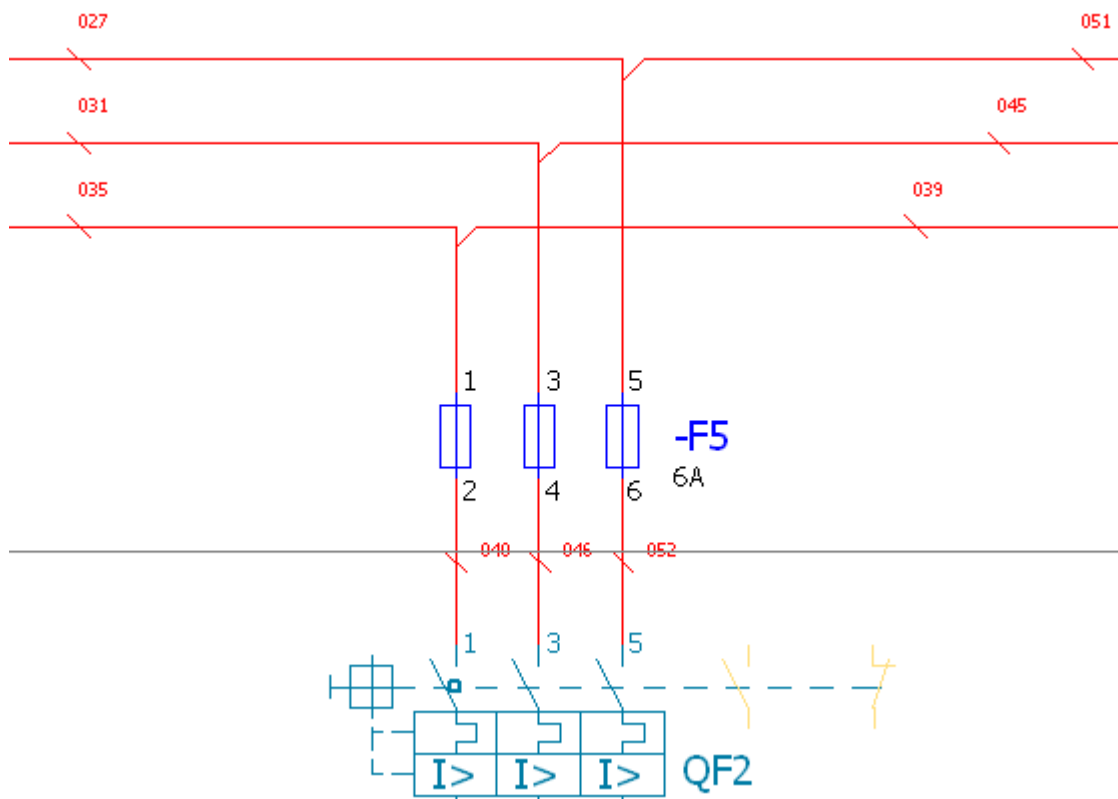


Рисунок 3.36 – Приклад автоматичної нумерації з'єднань

Для зручності монтажу й обслуговування шафи в ній передбачається установка клемних колодок. При цьому для з'єднання пристроїв уведення й виводу використовуються різні клемники, що відрізняються як способом з'єднання, так і діаметрами проводів, які приєднуються.

З обліком викладеного на схемі з'єднань слід передбачити мінімум два різних клемника.

При вставці клемника бажано, щоб клеми розташовувалися на одній лінії, хоча це не принципово. Для вставки клем і призначення їх властивостей виконайте наступне.

1. Виберіть пункт меню **Вставити > Символ**. Відкриється діалогове вікно **Выбор символа** (рис. 3.37).

2. Виберіть у діалоговому вікні **Выбор символа** вкладку **Дерево**. Відкриється дерево символів, у якому розкрийте список **Клемма**, як показано на рисунку 3.37.

3. Виділіть в списку клему й клацніть **ОК**. Символ клемки з'явиться поруч із курсором миші.

4. Поставте клему на крайню зліва лінію з'єднань. Натиснувши ліву кнопку миші, протягніть покажчик миші через усі з'єднання й відпустіть ліву кнопку миші. На рисунку 3.38 показано результат цієї операції – лінія клемника до відпускання кнопки миші з розставленими на лініях з'єднань клемками й позначенням у першій клемі X?.

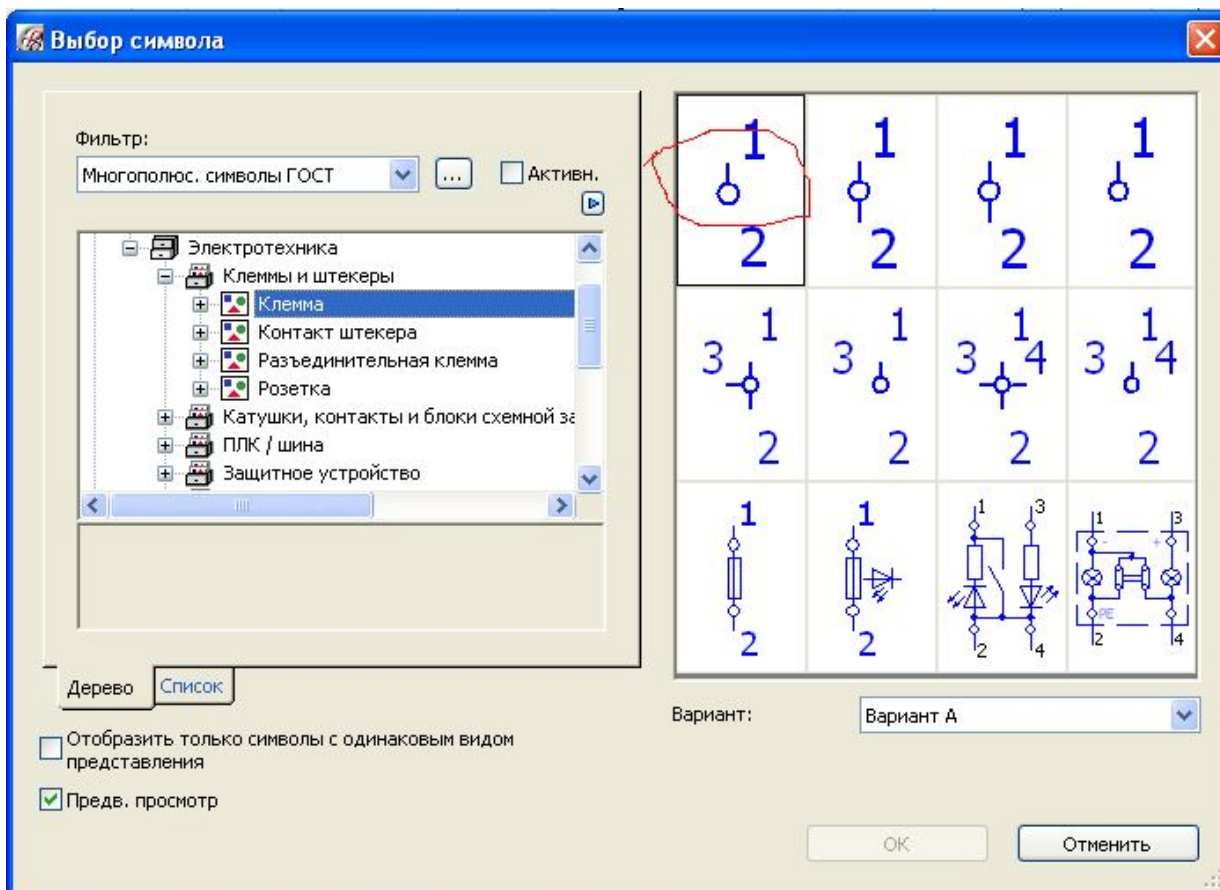


Рисунок 3.37 – Вид вікна **Выбор символа** на вкладці **Клемма**

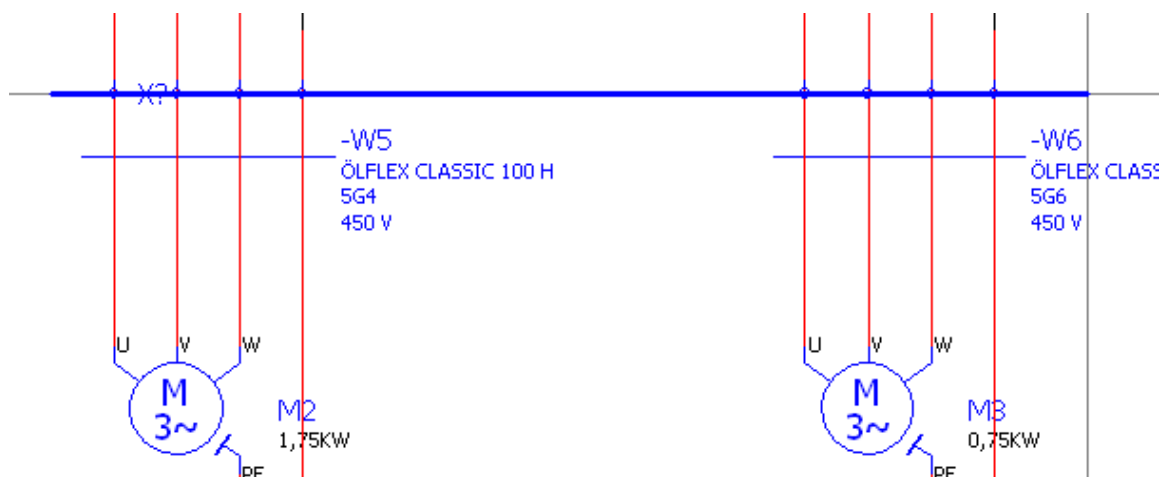


Рисунок 3.38 – Вид лінії клемника до відпускання кнопки миші

5. Відпустіть кнопку миші. З'явиться нумерація клем на лініях з'єднань, а також позначення нового клемника – X7 (у проекті вже встановлено 6 клем). Результат показано на рисунку 3.39. Тут клемник X7 застосований для приєднання кабелів W5 і W6.

6. Перервіть операцію установки клемника, виділіть клемник і в контекстному меню виберіть команду **Свойства**. Відкриється вікно **Свойства (усл. обозначение): Клеммы** яке показано на рисунку 3.40.

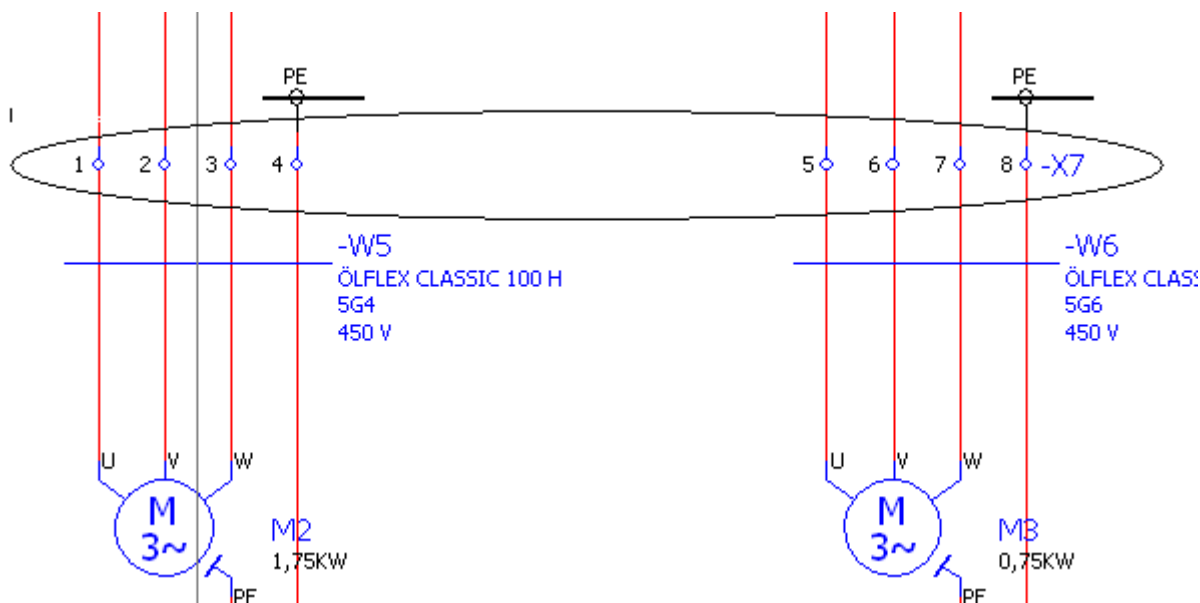


Рисунок 3.39- Вид клемника після відпускання кнопки миші

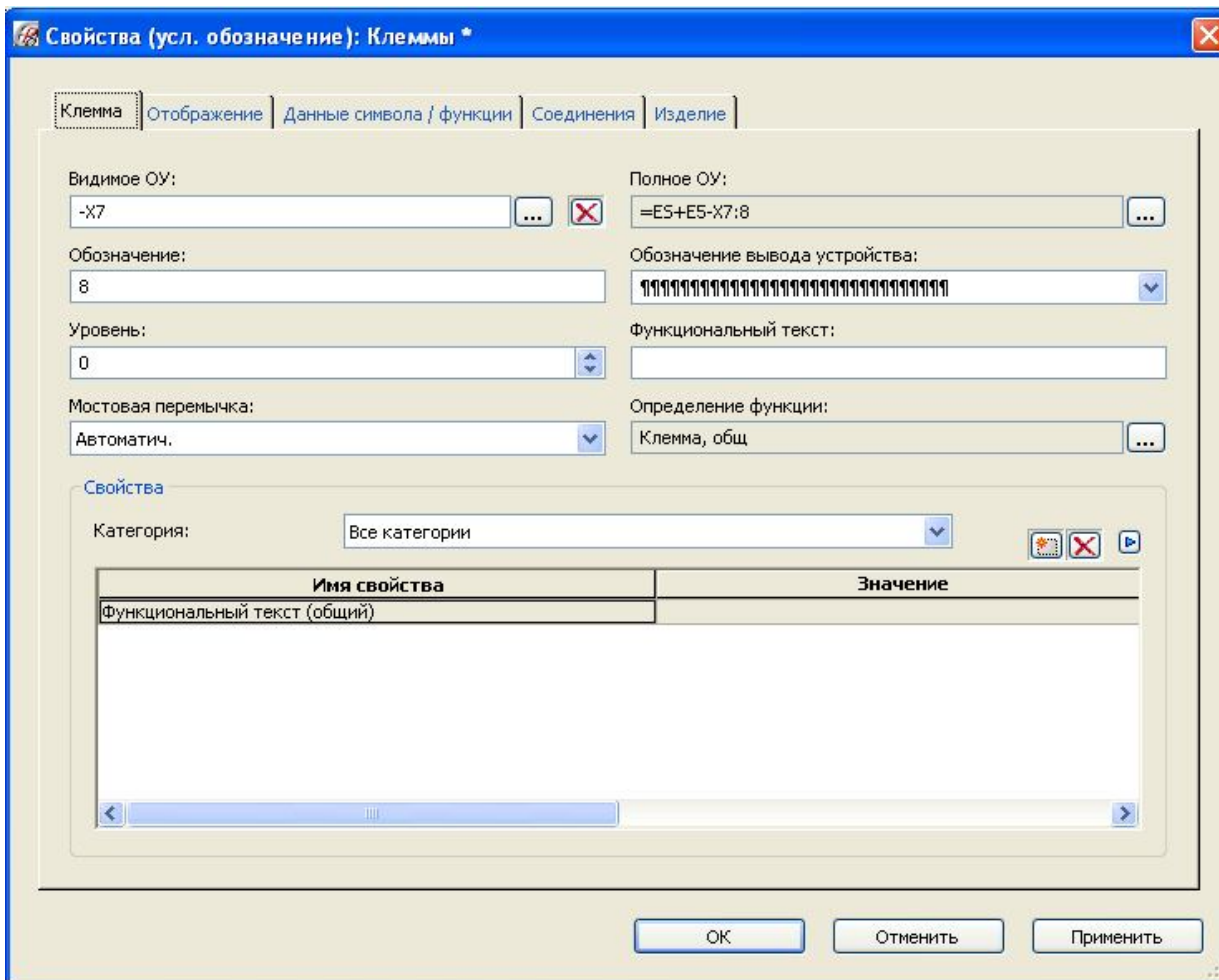


Рисунок 3.40 – Вид вікна *Свойства: Клеммы* на вкладці *Клемма*

У цьому вікні є 5 вкладок. На вкладці **Клемма** можна призначити видимий пристрій (тут клемма позначена «X7», але його можна

перенумерувати), а також позначення клеми (тут виділена клема 8, це позначення можна змінити, наприклад, на PE). У полі **Полное ОУ** повинна стояти повна ідентифікаційна адреса пристрою (тут =ES+E5-X7).

7. На вкладці **Отображение** виберіть *стандартну* схему порядку властивостей і в полі **Присоединение по умолчанию** установіть «Праворуч».

8. На вкладці **Изделие** клацніть лівою кнопкою в гнізді **Строка 1 – Номер изделия**. При цьому відкриється вікно **Выбор изделия**, у якому знайдіть список клем.

9. Виберіть підходящу клему, наприклад, для підключення двигунів підійде прохідний блок затискачів ENT.11511811 типу М6/8 з можливістю закріплення жил кабелю перетином до 8 мм². Коротка інформація про клему доступна в правій секції вікна (рис. 3.41). Клацніть на **ОК**. У вікні **Свойства** на вкладці **Клемма** з'явиться номер виробу і його докладні дані (рис. 3.42).

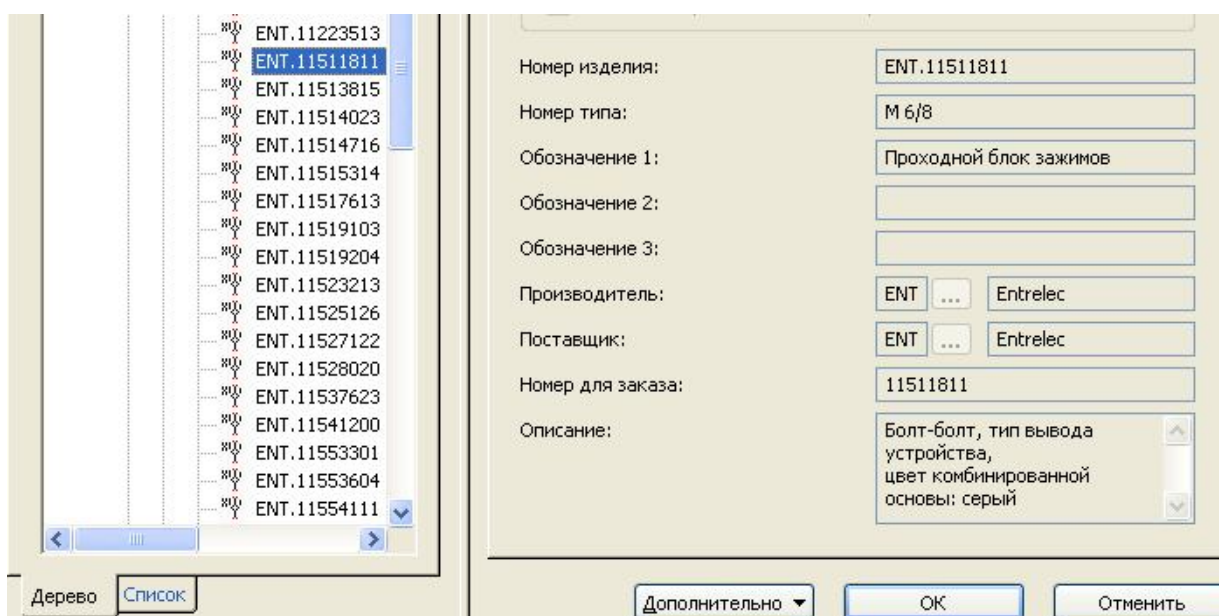


Рисунок 3.41 – Вибір клемника у вікні *Свойства* на вкладці *Изделие*

На цьому робота з розміщення клеми й призначенню властивостей клеми завершується. (Призначення властивостей можна також виконати командою **Определение клемника**).

3.5 Розміщення котушок контакторів і контактів

Нехай, наприклад, контактор використовується для включення двигуна. Нехай також котушка контактора повинна бути включена в деякий ланцюг керування напругою 24 В – у модуль виводу контролера з точкою підключення U2:1. Для вставки котушки контактора застосуємо команду **Вставить символ**. У вікні **Выбор символа** виберемо котушку типу К (рис. 3.43) і вставимо її в схему, як показано на рисунку 3.44.

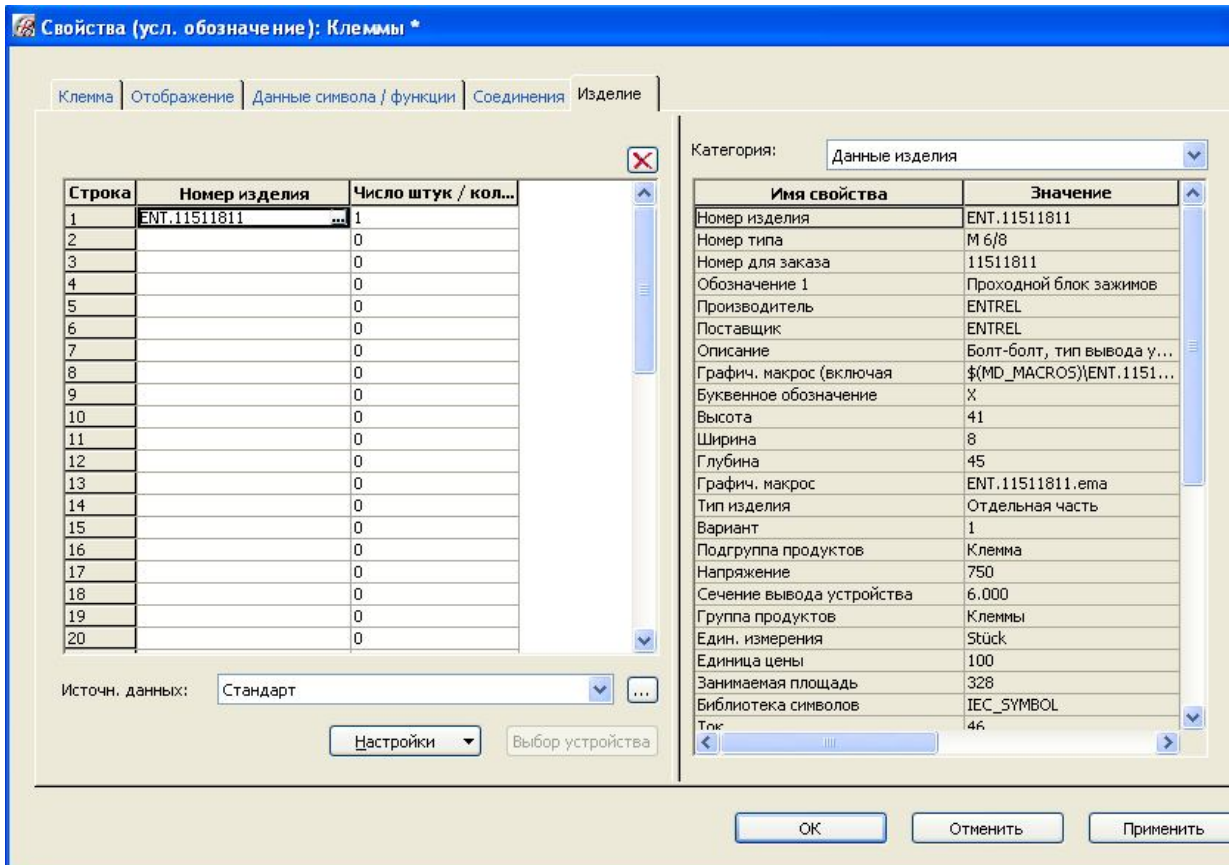


Рисунок 3.42 – Докладні дані про клему на вкладці **Изделие**

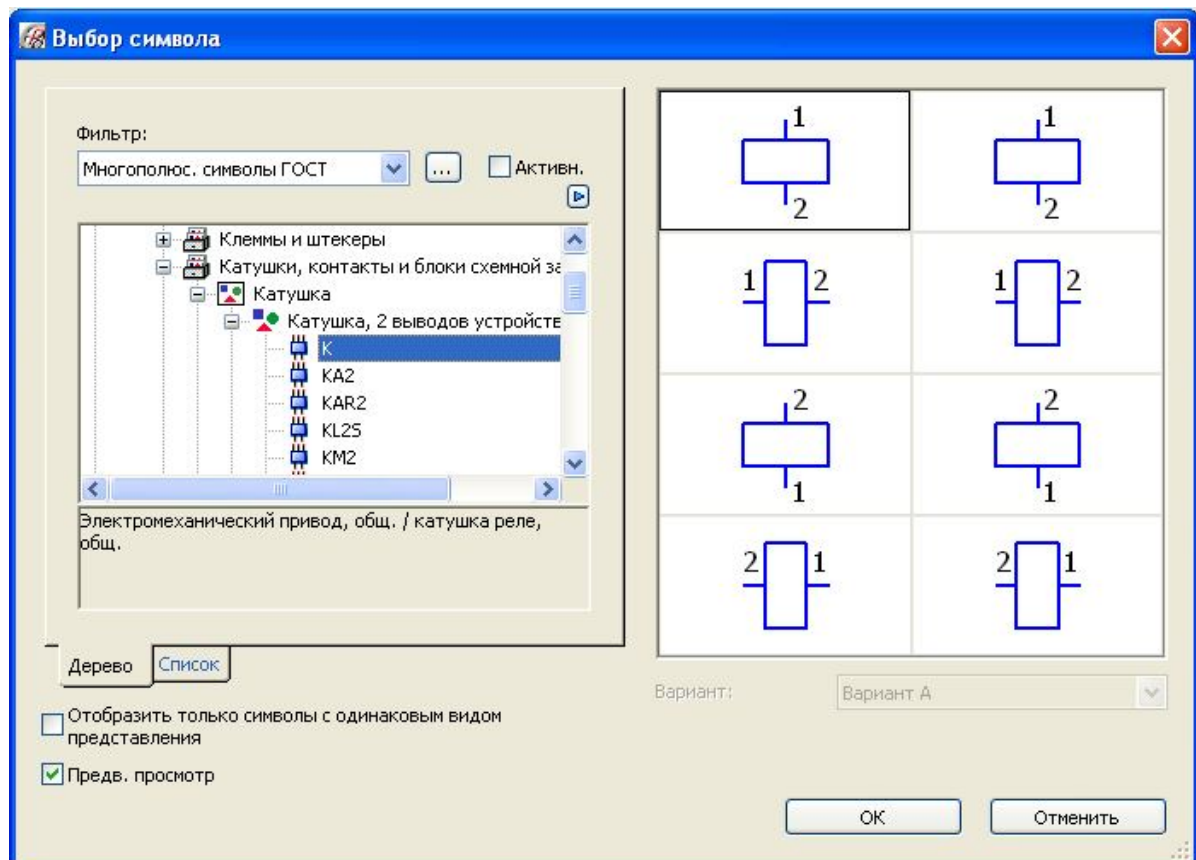


Рисунок 3.43 – Вибір символу катушки у вікні **Выбор символа**

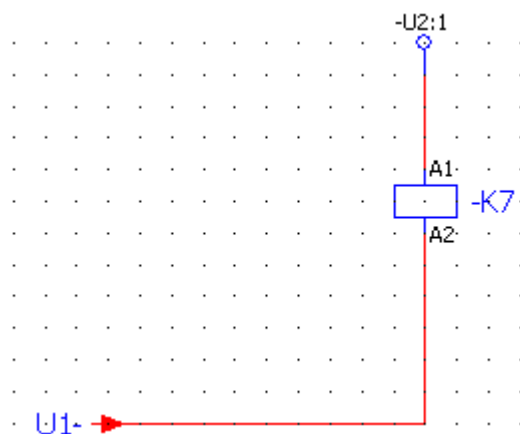


Рисунок 3.44 – Схема керування з котушкою контактора

Виділимо котушку, застосуємо команду **Свойства** контекстного меню й відкриємо вікно **Свойства...** На вкладці **Катушка** в поле **Свойства** натискаємо кнопку **Создать** (Ctrl+Alt+N) і у вікні, що відкрилося, знаходимо властивість «Напряжение катушки». Виділяємо цю властивість і тиснемо ОК. На вкладці **Катушка** в колонку «Имя свойства» з'являється властивість «Напряжение катушки». У стовпець **Значение** вписуємо «24 В». Результат цієї роботи показано на рисунку 3.45.

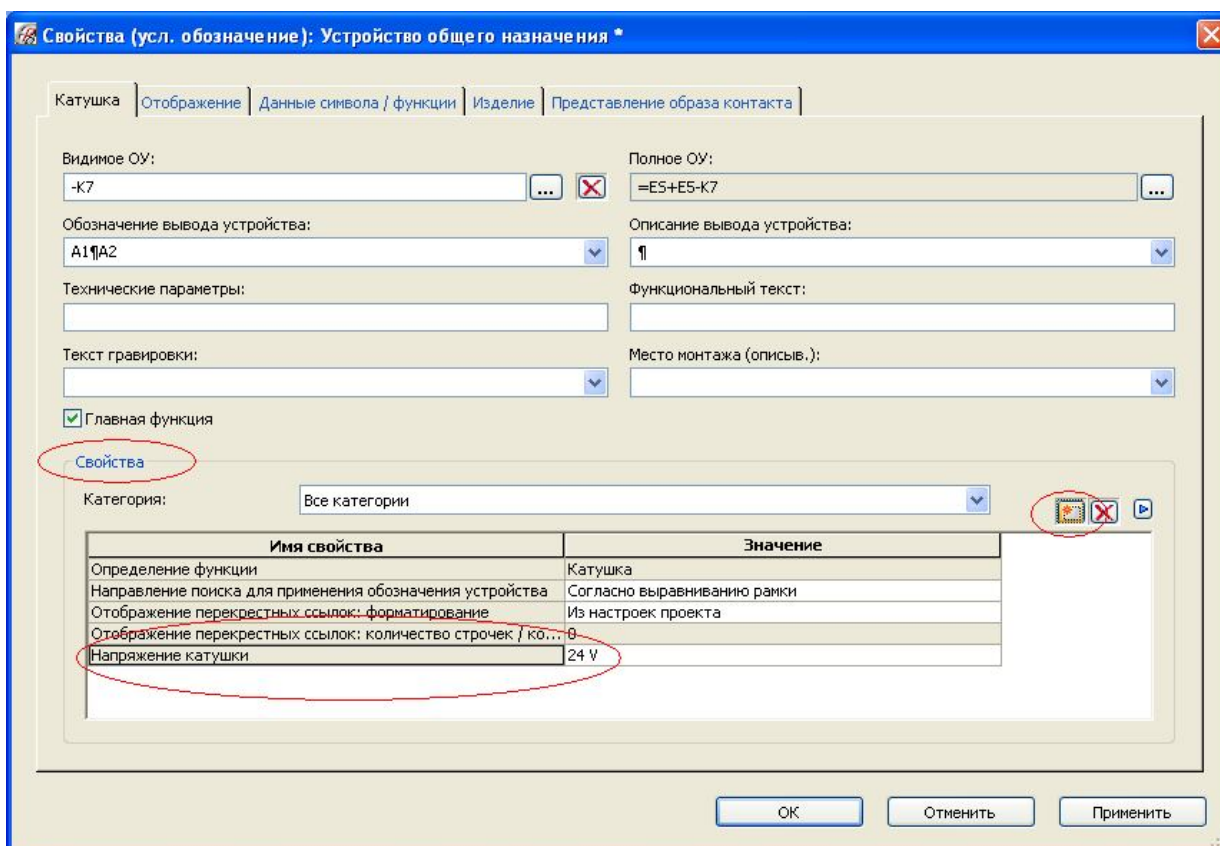


Рисунок 3.45 – Додавання властивості «Напряжение катушки»

На вкладці **Отображение** встановлюємо стандартну схему порядку властивостей і перемикаємося на вкладку **Изделие**. Тут тиснемо на кнопку **Выбор устройства**. Відкривається вікно **Выбор устройства** (рис. 3.46).

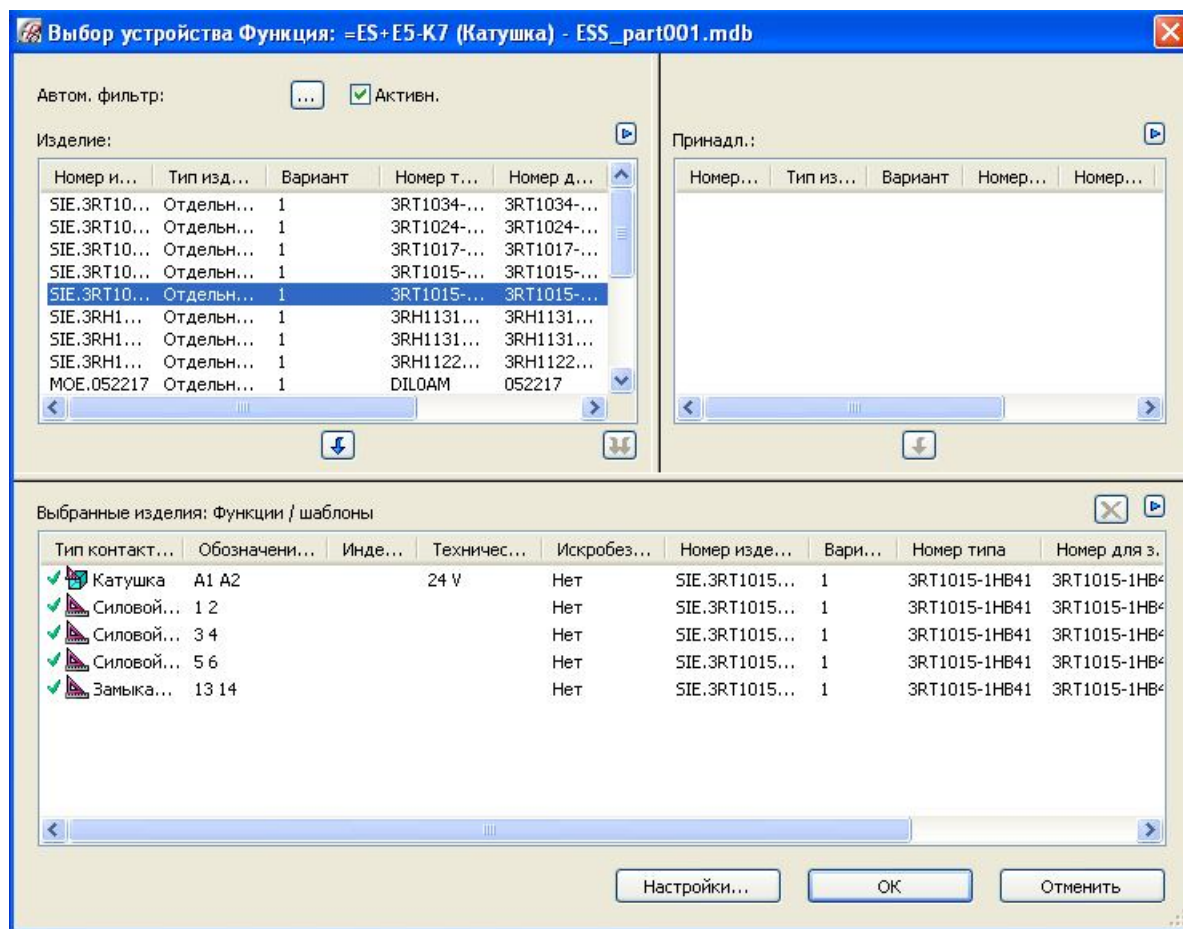


Рисунок 3.46 – Список контакторів з напругою 24 В

У полі **Изделие** цього вікна є список контакторів, відібраних редактором по напрузі котушки. У нижній частині вікна в полі **Выбранные изделия** наводяться дані по вибраному контактору.

З урахуванням функціонального призначення вибираємо контактор SIE.3RT1015-1HB41 із трьома силовими контактами й одним додатковим нормально розімкнутим контактом. Підтверджуємо вибір кнопкою **ОК** і автоматично вертаємося на вкладку **Свойства** (рис. 3.47).

Тут обраний контактор уже записано в рядку 1. Натискаємо кнопку **Применить** – усі дані контактора будуть відображені у правому полі вікна (призначена категорія «Данные изделия»). Закриваємо вікно кнопкою **ОК**.

Після призначення контактора його відображення на схемі (рис. 3.48) доповнюється вказівкою напруги живлення й умовними зображеннями контактів, які *прикріплюються* в нижній частині сторінки під зображенням котушки.

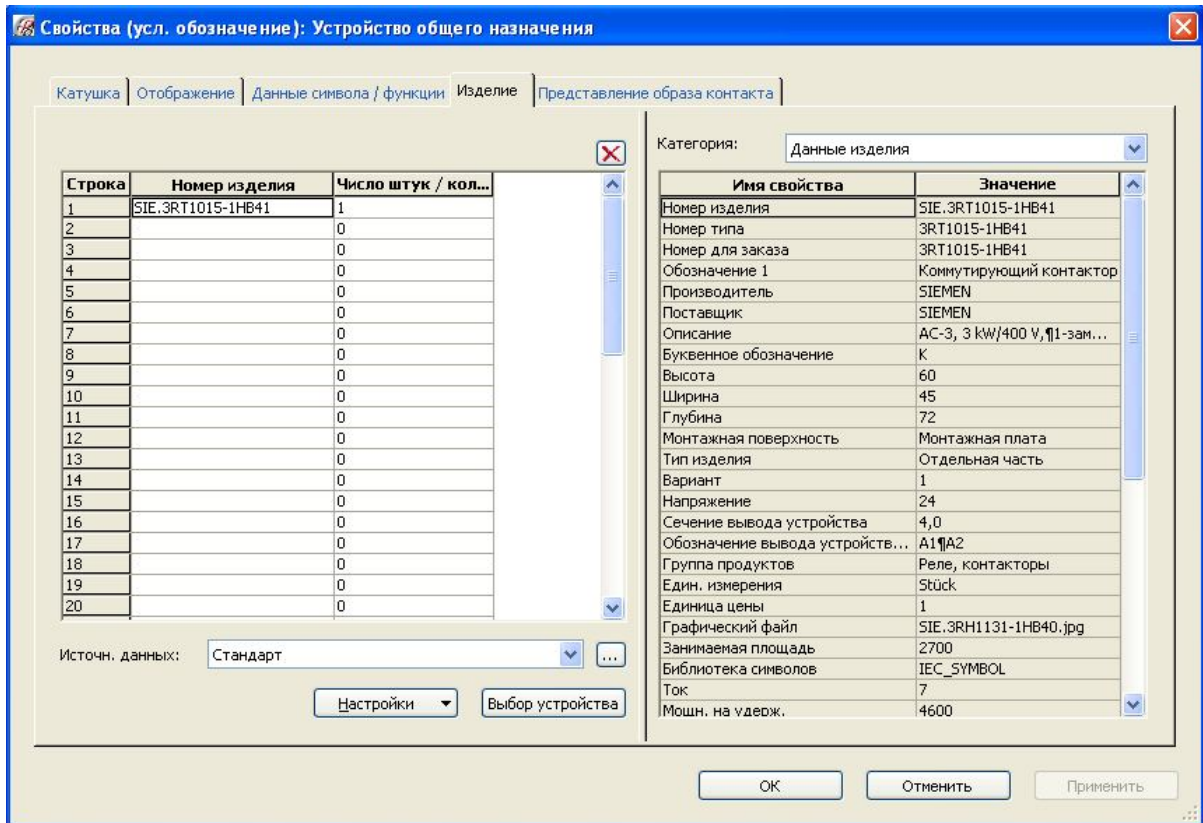


Рисунок 3.47 – Дані обраного контактора

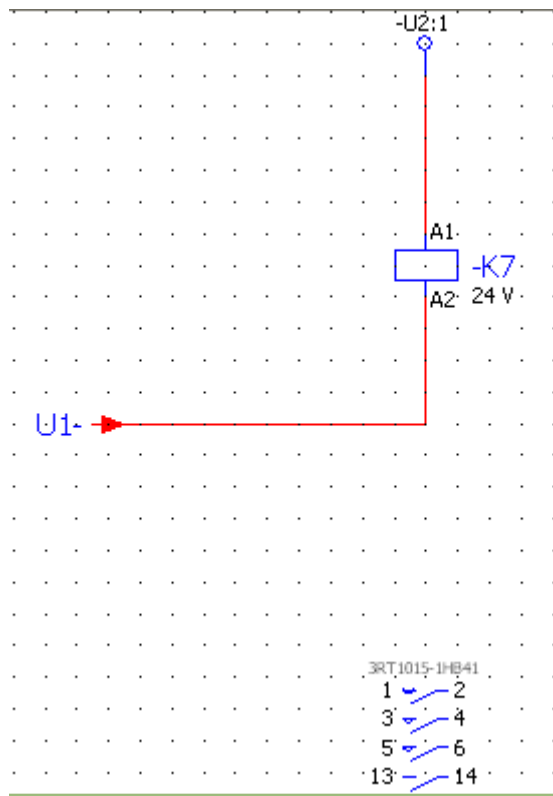


Рисунок 3.48 – Відображення контактора на схемі

Наступним етапом є розташування контактів у ланцюзі управління двигуна.

Застосуйте команду **Данные проекта ► Устройство ► Навигатор**. Відкриється вкладка **Устройство** «Імя проекту», що показана на рисунку 3.49. Ця вкладка являє собою навігатор пристроїв проекту.

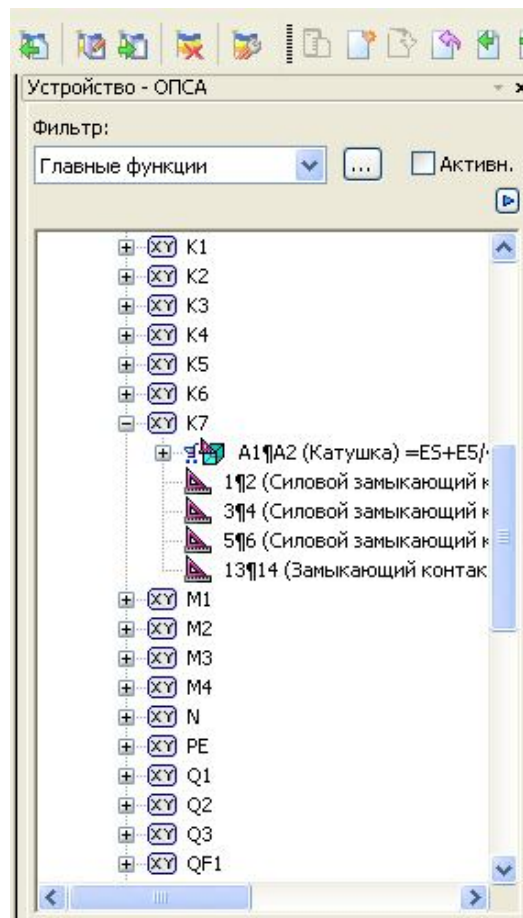


Рисунок 3.49 – Розгорнута вистава котушки K7 у навігаторові

У навігаторові втримуються всі пристрої, у тому числі й установлений контактор K7. У розгорнутій виставі видно, що котушка K7 установлена (іконка пофарбована зеленою призмою, а також зазначений ідентифікатор котушки), але контакти не встановлені (іконки контактів пофарбовані бордовим трикутником, ідентифікаторів немає).

Утримуючи клавішу **Ctrl**, виділимо три силові контакти контактора K7, правою кнопкою відкриємо контекстне меню й виберемо команду **Разместить ► Многополюсный**. По цій команді до курсору прикріплюється контакт. Розташовуємо контакт ліворуч від ліній з'єднань двигуна й, натиснувши ліву кнопку миші, перетнемо лінії з'єднань курсором, після чого відпускаємо кнопку миші.

Результат виконаних операцій показано на рисунку 3.50.

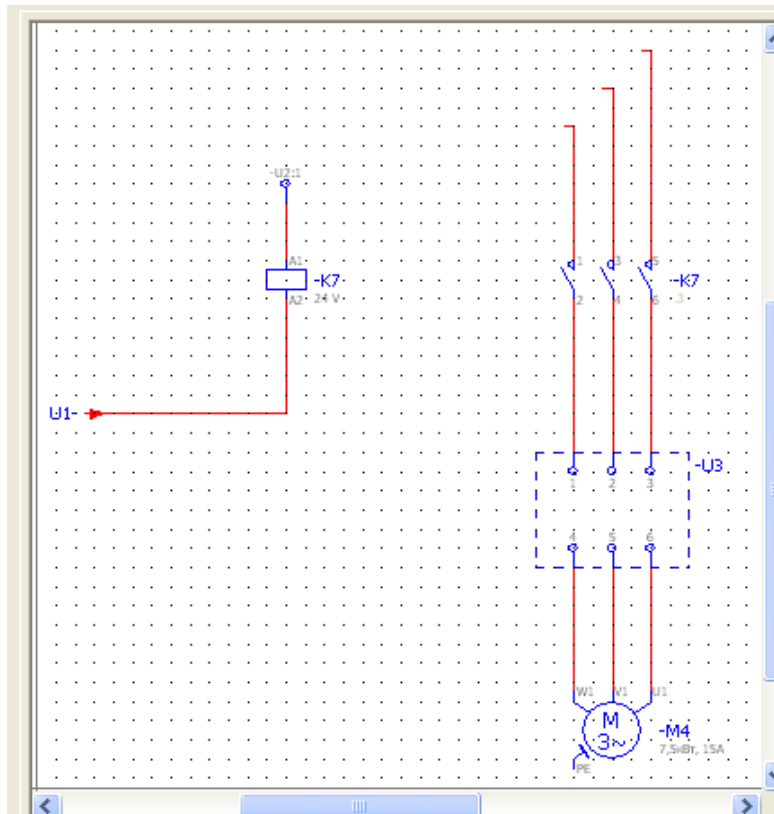


Рисунок 3.50 – Результат розміщення контактів у ланцюзі двигуна

3.6 Створення звітної документації проекту

EPLAN дозволяє створювати 18 типів звітів. Для практикуму актуальними є три типи звітів:

- ✓ Групова специфікація (перелік елементів).
- ✓ Список позначень пристроїв.
- ✓ Список (таблиця) з'єднань.

Процес створення цих звітів досить простий – застосуйте команду **Сервисные программы ► Отчеты ► Генерировать...**

По цій команді відкриється вікно **Отчеты**, показане на рисунку 3.51. Тут є дві вкладки – **Отчеты** та **Шаблоны**. На вкладці **Отчеты** слід вилучити «Титульный лист», тому що його згенерована форма не містить релевантної інформації. Титульний аркуш проекту слід скласти самостійно.

Генерація звіту починається з натискання кнопки **Создать**.

По завершенню процесу у навігаторові сторінок проекту з'явиться розділ REPORT із двома частинами – списками специфікацій і списками з'єднань (рис. 3.52)

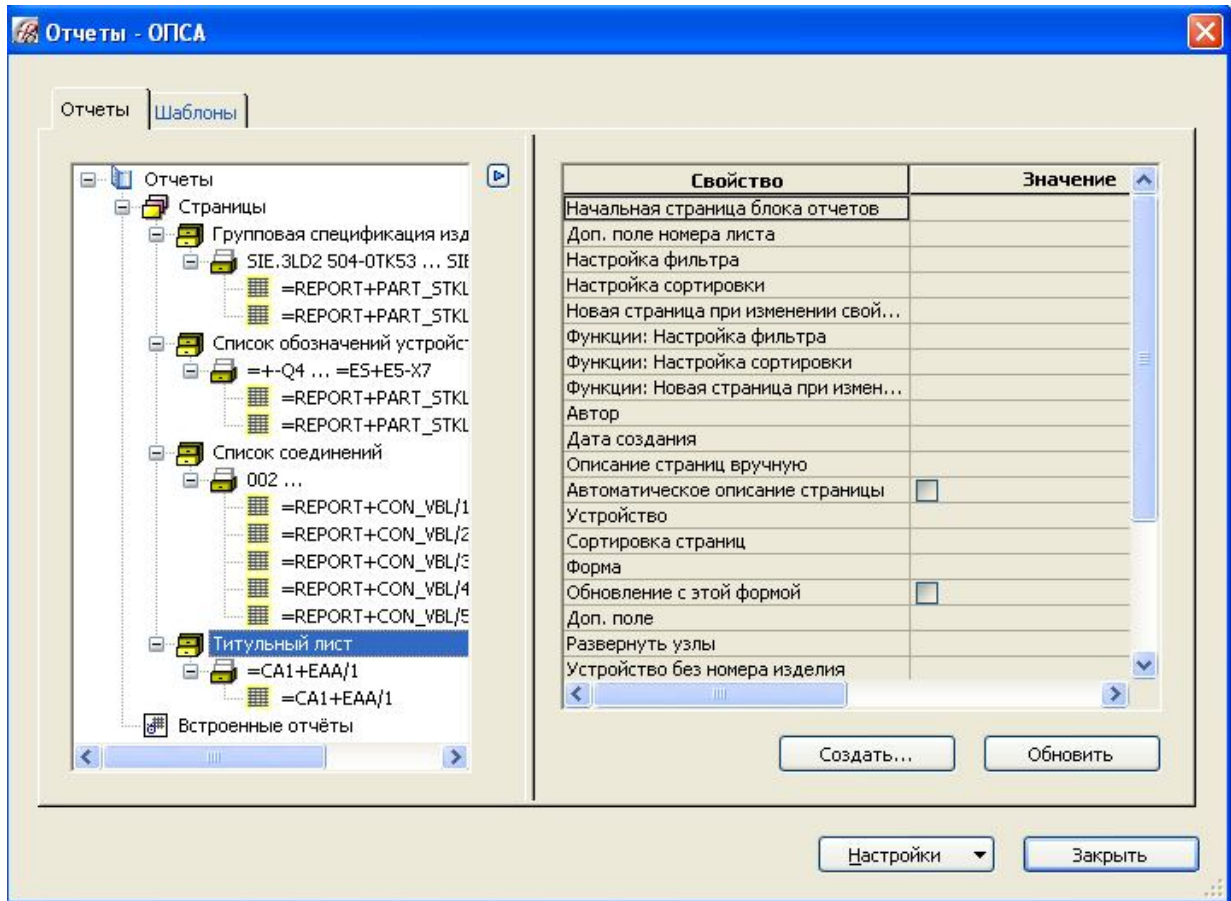


Рисунок 3.51 – Вікно **Отчеты** з відображенням усіх сторінок

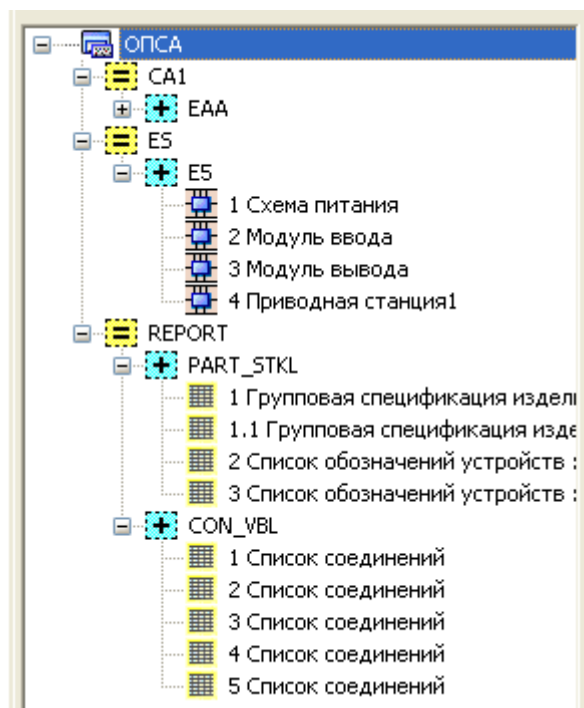


Рисунок 3.52 – Вид навігатора сторінок з відображенням схем і звітів

Виконаний проект можна експортувати у формат PDF. Для цього в навігаторі проекту виділіть ім'я проекту, відкрийте меню **Страница** й

виберіть **Экспортировать ► PDF...**. При цьому відкриється вікно **Экспорт PDF**, у якому потрібно вказати адресу для збереження pdf-файлу.

Графічна частина може бути експортована у формат DWG, який використовується в AutoCAD.

Дані методичні вказівки складені як уведення в EPLAN. Багато функцій і налаштувань залишилися в них не освітленими. У випадку утруднень із реалізацією деякого завдання проектування слід використовувати довідкову систему EPLAN, яку можна викликати командою **F1** або в меню **Справка**.

3.7 Вимоги до оформлення й захисту звіту

Сгенерований EPLAN звіт по цій роботі має 14 сторінок. Для печатки звіту можна використовувати два варіанти:

1. Застосувати команду **Проект ► Печать**.
2. Перетворити проект у формат PDF і потім роздрукувати отриманий файл.

Перед печаткою звіту необхідно переконатися в наступному:

1. *На титульному аркуші* присутні всі необхідні відомості, основний напис має номер документа, найменування проекту, прізвища автора й викладача.

2. *На схемах* основний напис містить усі відомості, номер документа є як в основному написі, так і в протилежному куті з розворотом 180°. Схеми виконані відповідно до вимог стандартів. Позиційні позначення мають правильне розміщення (праворуч або зверху від елемента), їхня нумерація розташована вірно (зліва направо, зверху вниз). Є перехресні посилання.

3. *У документації по проекту* основний напис на всіх аркушах оформлений. У груповій специфікації кількість записаних елементів по кожній групі відповідає кількості елементів, розміщених на схемах. У списках з'єднань усі проведення (крім проводів схеми підключень) мають номери, для всіх з'єднань установлені позначення елементів і їх виводів (звідки йде й куди надходить).

Якщо в процесі перевірки виявлені неточності й недогляди, то вони повинні бути виправлені. Після виправлення схем необхідно виправити звітну документацію. Для цього слід виділити розділ звіту (REPORT) і потім вибрати команду **Сервисные программы ► Отчет ► Обновить**.

Слід урахувати, що після відновлення звіту основний напис на сторінках звіту прийдеється виконувати заново.

При захисті звіту необхідно відповісти на запитання, що ставляться до вимог стандартів при створенні принципових електричних схем, схем підключень і з'єднань.

4 РОЗРОБКА ТЕКСТОВИХ МАТЕРІАЛІВ ПРОЕКТУ

Ціль роботи: придбати вміння й навички в створенні текстових документів проекту засобами Windows-додатків.

4.1 Основні вимоги до викладу й оформленню текстових документів

При оформленні текстового документа слід керуватися стандартом ДСТУ 3008-95. ДОКУМЕНТАЦІЯ. ЗВІТИ В СФЕРІ НАУКИ Й ТЕХНІКИ.

Звіт оформляють на аркушах формату А4 (210 x 297 мм).

Текст звіту слід оформляти, дотримуючи наступні розмірів полів: верхнє, ліве й нижнє – не менш 20 мм, праве – не менш 10 мм.

Основним шрифтом є шрифт Times New Roman 14.

Абзацний відступ повинен бути однаковим по всьому тексту звіту й рівним п'ятьом знакам.

Заголовки структурних елементів звіту й заголовки розділів слід розташовувати в середині рядка й друкувати прописними буквами без крапки наприкінці, не підкреслюючи.

Заголовки підрозділів, пунктів і підпунктів звіту слід починати з абзацного відступу й друкувати малими літерами, крім першої прописної, не підкреслюючи, без крапки наприкінці. Якщо заголовок складається із двох або більш речень, їх розділяють крапкою. Переноси слів у заголовку не допускаються.

Відстань між заголовком і наступним або попереднім текстом повинна бути не менш трьох інтервалів. Відстань між підставами рядків заголовка, а також між двома заголовками ухвалюють такою же, як у тексті.

Не допускається розміщати найменування розділу, підрозділу, а також пункту й підпункту в нижній частині сторінки, якщо після нього розташований тільки один рядок тексту.

Сторінки звіту слід нумерувати арабськими цифрами, дотримуючи наскрізну нумерації по всьому тексту звіту. Номер сторінки проставляють у правому верхньому або нижньому кутах сторінки без крапки наприкінці.

Титульний аркуш включають у загальну нумерацію сторінок звіту. Номер сторінки на титульному аркуші не проставляють.

Ілюстрації й таблиці, розташовані на окремих сторінках, включають у загальну нумерацію сторінок звіту.

Розділи підрозділи, пункти, підпункти звіту слід нумерувати арабськими цифрами.

Підрозділи повинні мати порядкову нумерацію в межах кожного розділу. Номер підрозділу складається з номера розділу й порядкового номера

підрозділу, які розділяються крапкою. Після номера підрозділу крапку не ставлять, наприклад: 1.1, 1.2 і т.д.

Номер пункту складається з номера розділу й порядкового номера пункту або з номера розділу, порядкового номера підрозділу й порядкового номера пункту, поділених крапкою. Після номера пункту крапку не ставлять, наприклад: 1.1, 1.2 або 1.1.1, 1.1.2 і т.д.

Ілюстрації (креслення, рисунки, графіки, схеми, діаграми, фотознімки) слід розташовувати у звіті безпосередньо після тексту, у якому вони згадуються вперше, або на наступній сторінці. На всі ілюстрації повинні бути дані посилання у звіті.

Якщо ілюстрації створені не автором звіту, необхідно, представляючи їх у звіті, дотримувати вимог чинного законодавства про авторські права.

Креслення, графіки, схеми, діаграми, поміщені у звіті, повинні відповідати вимогам стандартів «Єдина системи конструкторської документації» і «Єдина системи програмної документації».

Ілюстрації слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком ілюстрацій, що приводяться в додатках.

Номер ілюстрації складається з номера розділу й порядкового номера ілюстрації, які розділяються крапкою, наприклад: рисунок 3.2 – другий рисунок третього розділу.

Ілюстрація позначається словом «Рисунок ___», яке разом з назвою ілюстрації поміщають після пояснюючих даних, наприклад: «Рисунок 4.1 – Схема розміщення».

Таблиці застосовують для вистави цифрового або класифікаційного матеріалу.

Горизонтальні й вертикальні лінії, які розмежовують рядки таблиці, а також лінії, що обмежують таблицю зліва, справа і знизу, можна не проводити, якщо їх відсутність не утрудняє користування таблицею.

Таблицю слід розташовувати безпосередньо після тексту, у якому вона згадується вперше, або на наступній сторінці. На всі таблиці повинні бути посилання в тексті звіту.

Таблиці слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком таблиць, що приводяться в додатках.

Номер таблиці складається з номера розділу й порядкового номера таблиці, які розділяються крапкою, наприклад: таблиця 2.1 – перша таблиця другого розділу.

Таблиця може мати назву, яку друкують малими літерами (крім першої прописної) і поміщають над таблицею. Назва повинна бути короткою і відбивати зміст таблиці.

Якщо рядки або графи таблиці виходять за формат сторінки, таблицю ділять на частині, поміщаючи одну частину під іншою або поруч, або переносячи частину таблиці на наступну сторінку. При цьому в кожній частині таблиці повторюють її головку й боковик.

При розподілі таблиці на частині допускається її головку або боковик замінити, відповідно, номерами граф або рядків. При цьому нумерують арабськими цифрами графи й / або рядки першої частини таблиці.

Слово «Таблиця ___» указують один раз зліва над першою частиною таблиці, над іншими частинами пишуть «Продовження таблиці ___» із вказівкою номера таблиці.

Заголовки граф таблиці друкують із прописних букв, підзаголовки – з рядкових.

Підзаголовки, що мають самостійне значення, пишуть із прописної букви. Наприкінці заголовків і підзаголовків таблиць крапки не ставлять. Заголовки й підзаголовки граф указують в однині.

Формули й рівняння розташовують безпосередньо після тексту, у якому вони згадуються, посередині сторінки. Вище й нижче кожної формули або рівняння повинно бути пропущено не менш одному рядка.

Формули й рівняння у звіті (за винятком формул і рівнянь, наведених у додатку) слід нумерувати порядковою нумерацією в межах розділу.

Номер формули або рівняння складається з номера розділу й порядкового номера формули або рівняння, розділених крапкою, наприклад: формула (1.3) – третя формула першого розділу.

Номер формули або рівняння вказують на рівні формули або рівняння в дужках у крайньому правому положенні на рядку.

Математичні символи у формулах і тексті необхідно представляти в загальноприйнятому стилі Math, Matrix-Vector і ін.

Пояснення значень символів і числових коефіцієнтів, що входять у формулу або рівняння, слід приводити безпосередньо під формулою в тій послідовності, у якій вони дані у формулі або рівнянні.

Пояснення значення кожного символу й числового коефіцієнта слід давати з нового рядка. Перший рядок пояснення починають із абзацу словом «де» без двокрапки, наприклад:

«Відомо, що

$$Z = \frac{M_1 - M_2}{f_1 + f_2}, \quad (3.1)$$

де M_1, M_2 – математичні очікування;

f_1, f_2 – середньоквадратичне відхилення міцності й навантаження [23]».

Переносити формули або рівняння на наступний рядок допускається тільки на знаках виконуваних операцій, причому знак операції на початку наступного рядка повторюють. При переносі формули або рівняння на знаку операції множення застосовують знак «×».

Додатки слід оформляти як продовження звіту на його наступних сторінках або у вигляді окремої частини, розташовуючи додатки в порядку появи посилань на них у тексті звіту.

Якщо додатки оформляють на наступних сторінках звіту, кожний додаток повинен починатися з нової сторінки. Додаток повинен мати заголовок, надрукований угорі малими літерами з першої прописний симетрично щодо тексту сторінки. Посередині рядка над заголовком малими літерами з першої прописної повинно бути надруковане слово «Додаток ___» і прописна буква, що позначає додаток.

Додатки слід позначати послідовно прописними буквами українського алфавіту, за винятком І, З, Ї, Й, О, Ч, Ь, наприклад: додаток А, додаток Б і т. д.

Один додаток позначається як додаток А.

Додатки повинні мати загальну з іншою частиною звіту наскрізну нумерацію сторінок.

Наявні в тексті додатка ілюстрації, таблиці, формули й рівняння слід нумерувати в межах кожного додатка, наприклад: Рисунок Г.3 – третій рисунок додатка Г; Таблиця А.2 – друга таблиця додатка А; формула (А.1) – перша формула додатка А.

Якщо в додатку одна ілюстрація, одна таблиця, одна формула, одне рівняння, їх нумерують, наприклад: Рисунок А.1, Таблиця А.1, формула (В.1).

4.2 Методика виконання роботи, зміст звіту і його захист

Для самостійного виконання роботи кожний студент одержує індивідуальне завдання (рисунок, формули, таблиці) в обсязі 4-10 сторінок.

Завдання виконується в наступній послідовності:

- 1 Із застосуванням Word створюється текстова частина документа.
- 2 Із застосуванням Visio виконуються рисунки.
- 3 Оформляються таблиці документа (Word, Excel).

4 Рисунки й таблиці вставляються в текст.

5 Проводиться форматування документа і його оформлення.

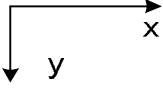
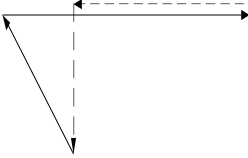
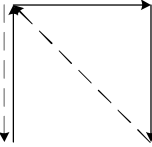
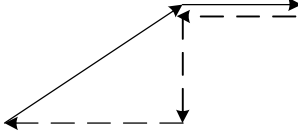
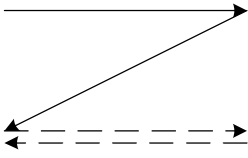
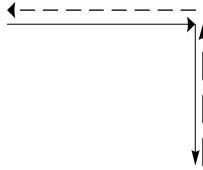
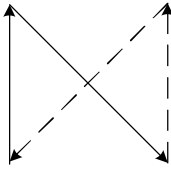
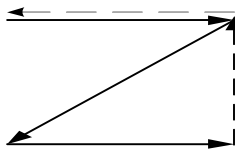
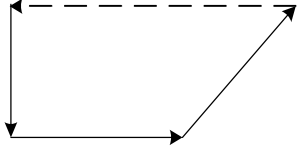
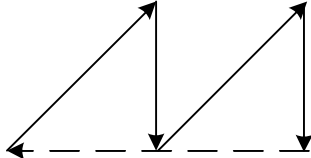
Звіт повинен містити титульний аркуш, основний напис на сторінці 2 і текстовий матеріал в обсязі не менш 6 сторінок.

При захисті звіту враховуються якість оформлення, дотримання вимог стандарту, наявність граматичних помилок, а також якість відповідей на контрольні питання.

ЛІТЕРАТУРА

1. Петров И.В. Програмируемые контроллеры. Стандартные языки и инструменты / Под ред. проф. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2003. – 256 с.
2. Руководство пользователя по программированию ПЛК в CoDeSys 2.3. Русская редакция: ПК Пролог, Россия.
3. Бернд Гишель. EPLAN Electric P8. Практическое пособие пользователя/ EPLAN Software & Service Россия, 2010 – 509 с.
4. Интернет посилання:
 - 3S Smart Software Solutions
<http://www.3s-software.com>
 - ПК «Пролог»
<http://www.prolog.smolensk.ru>
 - www.eplan-russia.ru/

Додаток А. Варіанти завдань до роботи 1

Вар.	Вихідні дані				Положення системи координат 
	X1	X2	Y1	Y2	
1	-30	40	-30	30	
2	50	-50	-50	50	
3	60	40	-40	40	
4	60	-60	40		
5	80		40		
6	70	-70	-70	70	
7	80		80		
8	40	30	30		
9	30	30	30	-30	

10	40	40	40	-40	
11	90	-40	30	30	
12	80	-30	50		
13	-20	50	-20		
14	30		30		
15	30	20	-30		
16	40		40		
17	50		50		

18	30		-30		
19	60		60		
20	20	20	20		
21	60	60	-60		
22	50	30	-50	-30	
23	80	-40	20	20	
24	70	-20	50		
25	-15	45	-15		

ЗМІСТ

1 ВИВЧЕННЯ СИСТЕМИ ПРОГРАМУВАННЯ КОНТРОЛЕРІВ CoDeSys..	3
1.1 Головне вікно системи CoDeSys.....	3
1.2 Компонентна організація проекту	5
1.3 Настроювання опцій проекту.....	10
1.4 Команди керування проектом і об'єктами.....	15
1.5 Приклад створення проекту в CoDeSys	17
1.6 Методика візуалізації проекту	25
1.7 Варіанти завдань і зміст звіту по роботі.....	28
2 РОЗРОБКА ПРОГРАМИ В ГРАФІЧНОМУ РЕДАКТОРІ LD.....	29
2.1 Методика роботи в графічному редакторі мови LD	29
2.2 Приклад створення програми мовою LD	36
2.3 Варіанти завдань і зміст звіту по роботі.....	40
3 ЕЛЕКТРОТЕХНІЧНЕ ПРОЕКТУВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ в EPLAN ELECTRIC P8	41
3.1 Призначення програмного пакета EPLAN.....	41
3.2 Завдання проектування й рекомендації зі створення проекту	43
3.3 Створення принципів схем і схем підключень	50
3.4 Створення схем з'єднань	56
3.5 Розміщення котушок контакторів і контактів.....	74
3.6 Створення звітної документації проекту	80
3.7 Вимоги до оформлення й захисту звіту	82
4 РОЗРОБКА ТЕКСТОВИХ МАТЕРІАЛІВ ПРОЕКТУ	83
4.1 Основні вимоги до викладу й оформленню текстових документів.....	83
4.2 Методика виконання роботи, зміст звіту і його захист	86
ЛІТЕРАТУРА	87
ДОДАТОК А. Варіанти завдань до роботи 1	88

