Міністерство освіти і науки України
Запорізька державна інженерна академія

ПРОГРАМУВАННЯ СИСТЕМ РЕАЛЬНОГО ЧАСУ

Методичні вказівки
до лабораторного практикуму, самостійних і контрольних робіт

для студентів ЗДІА
спеціальності 6.050201
“Автоматизоване управління технологічними процесами”

Рекомендовано до видання
на засіданні кафедри АУТП,
протокол № 10 від11.01 .2017 р.

Запоріжжя
2017

Програмування систем реального часу: методичні вказівки до лабораторного практикуму, самостійних і контрольних робіт для студентів ЗДІА спеціальності 6.050201 “Автоматизоване управління технологічними процесами” /Ніколаєнко А.М.; Запоріз. держ. інж. акад. – Запоріжжя: ЗДІА, 2017.– 58с.

Укладач: А. М. Ніколаєнко, к.т.н., професор

Відповідальний за випуск : *зав. кафедри АУТП професор* ***М.Ю. Пазюк***

Рецензент:

Н.П. Полякова*, к.т.н., доцент кафедри програмного забезпечення автоматизованих систем Запорізької державної інженерної академії*

ЗМІСТ

Передмова

У даних методичних вказівках розроблено лабораторний практикум для отримання навичків програмування логічних контролерів мовами міжнародного стандарту IEC 61131-3, серед яких три графічних (FBD, SFC, LD), орієнтованих на інженерів і дві текстові (ST, IL), що орієнтовані на програмістів.

LD (Ladder Diagram – релейні діаграми) є стандартним варіантом класу графічних мов релейно-контактних схем. Логічні вирази цією мовою зображуються у вигляді реле. Під час програмування передбачено використання не лише базових елементів "контакт" і "котушка", але і функціональних блоків стандарту IEC 61131-3.

FBD (Functional Block Diagram - діаграми функціональних блоків) – графічна мова, схожа на LD, але замість реле використовує функціональні блоки.

IL (Instruction List - список команд) – текстова мова програмування низького рівня, синтаксично схожа на асемблер.

ST (Structured Text - структурований текст) – текстова мова високого рівня. По синтаксису схожа на Pascal. Самостійного значення майже не має, використовується спільно з SFC.

SFC (Sequential Function Chart - послідовні функціональні схеми) – графічна мова, яка описує алгоритм у вигляді набору зв'язаних пар "крок-перехід". Крок – це набір операцій над змінними, а перехід – набір виразів, що визначає передачу правління наступному кроку. SFC має можливість розпаралелювання алгоритму, але не має засобів для опису кроків і переходів, які можуть бути виражені тільки засобами інших мов стандарту.

Розроблення додатків користувача в лабораторних роботах здійснюється в Softlogic-системі програмування KW MULTIPROG, яка має функціональні можливості по налагодженню та діагностуванню програм, а також тестуванню їх роботи без участі контролера в режимі симулятора.

Лабораторна робота №1
Розробка проекту користувача в системі програмування MULTIPROG
і робота в редакторі проектного дерева

Розроблення проекту користувача

Для створення проекту користувача необхідно запустити систему програмування, двічі клацнувши по піктограм MULTIPROG на робочому столі ЕОМ. Використовуючи пункт меню File, відкрити діалогове вікно New Project, рис. 1.1.

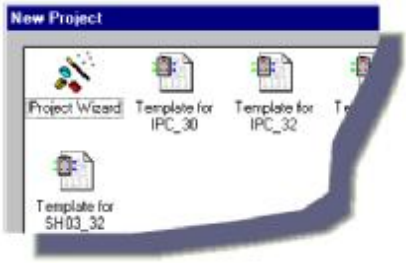


Рис. 1.1. Фрагмент діалогового вікна

Двічі клацнути по Project Wizard (Майстер проекту). У діалоговому вікні, що з'явиться, ввести ім'я проекту My_first_Project (Мій перший проект). Майстер проекту збереже проект у відповідному файлі My_first_Project .mwt і створить підтеку того ж імені, де будуть запам'ятовуватися тіло коду і файли змінних.

Максимальна довжина імені проекту може складати 24 символи і не повинна містити пропуски або спецсимволи.

За замовчуванням шлях проекту вводиться автоматично. Якщо бажаєте створити інший шлях, натисніть кнопку, на якій є три крапки і у діалозі Celect Directory (Вибрати директорію) виберіть теку для нового проекту, клацніть ОК.

Натисніть кнопку Next (Далі) і у рядок Name of POU (Імя POU) введіть ім'я першого організатора програми (POU) – Main (Головний). Оскільки зразковий проект будемо програмувати графічною LD-мовою активізуйте відповідний перемикач.

Натисніть кнопку Next (Далі) і введіть бажане ім'я конфігурації. У нашому прикладі це – Configuration (Конфігурація). У вікні зі списком виберіть тип конфігурації проекту IPC_40, тому що компілятор генерує intel- код для ProConOS 4.8.

Натисніть кнопку Next (Далі) і введіть ім'я ресурсу – Resource (Ресурс). У вікні зі списком виберіть тип ресурсу – PCOS_NT.

Натисніть кнопку Next (Далі) і введіть ім'я задачі –Task (Задача). У вікні зі списком виберіть тип задачі – Cyclic (Циклічний).

Натисніть кнопку Next (Далі). На екрані з'явиться зміст зроблених установок. Якщо помилок немає, натисніть Finish (Готов), в іншому разі скоригуйте кроки проекту, де є помилка, користуючись кнопкою „Back" (Назад).

Створений новий проект Main з'явиться у вікні проектного дерева, рис.1.2. Проект складається з бібліотеки (Libraries), типу даних (Data Type), програмних модулів (Logical POUs) і набору елементів конфігурації, які зображені у піддереві Physical Hardware (Фізична апаратура).

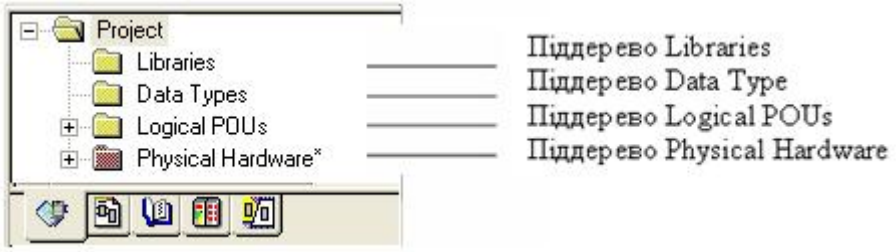


Рис.1.2 Дерево проекту

Теки і робота в редакторі проектного дерева

Тека бібліотеки (Libraries) – це POUs, що оголошені як бібліотеки. Існують бібліотеки мікропрограм і бібліотеки користувача.

Бібліотеки мікропрограм – це бібліотеки, які створені виробниками PLC. Файл такої бібліотеки має розширення *.fwl.

Бібліотеки, що призначені користувачем, – це проекти, функціональні блоки та функції, які створені раніше. Маючи вже закінчений проект, створений функціональний блок або функцію, можна використовувати їх POUs і робочі листки у нових проектах багато разів. Тобто користувачеві немає потреби створювати тіло коду, яке вже існує.

Бібліотека користувача знаходиться у файлі, з розширенням *.mwl.

При створенні бібліотеки потрібно дотримуватися наступного:

- для оголошення бібліотек можуть використовуватися тільки POUs;
- в оглядовому режимі робочі листки POUs оголошених бібліотек користувача можуть бути тільки показані і не можуть редагуватися;
- у той же час робочі листки бібліотеки користувача, що відкриті в оглядовому режимі, можуть перемикатися в оперативний режим, в якому можливе їх налагодження.

Щоб оголосити бібліотеку необхідно у проектному дереві позначити теку піддерева Libraries (Бібліотеки). В панелі інструментів клацнути по іконі Add Object (Додати об'єкт) або вибрати пункт меню Add Library (Додати бібліотеку) в підменю Project (Проект).

У діалоговому вікні Include library (Вставити бібліотеку), що з'явиться, у рядку «Папка» вибрати і відкрити теку MULTIPROG, а в теці Project (Проект) позначити файл з розширенням *.mwl, який бажано оголосити як бібліотеку, і у діалоговому вікні Include library (Включити бібліотеку) натиснути кнопку Include. Ікона створеної бібліотеки вставиться у проектне дерево.

Бібліотеки мають власне піддерево у проектному дереві. Можна показати повне проектне дерево або тільки піддерево "Бібліотеки", клацаючи по етикетці Libraries (Бібліотеки), що внизу вікна проектного дерева.

Піддерево Libraries (Бібліотеки), рис.1.3, складається з двох або більше ікон. Перша ікона – це вузол директорії. Елемент цього вузла директорії являє оголошену бібліотеку.

Рис. 1.3 Піддерево бібліотеки користувача

Примітка. Якщо при створенні бібліотеки ікона Add Object (Додати об'єкт) інертна, то доступ до бібліотеки захищений. Для отримання доступу необхідно ввести пароль, використовуючи пункт меню Enter password (Ввести пароль) в підменю File (Файл).

Для видалення бібліотеки треба позначити її ікону в проектному дереві і натиснути клавішу "DEL", з'явиться попередження, в якому потрібно підтвердити видалення.

Тека типу даних, які призначені користувачем, оголошуються у робочому листку типу даних, обов'язково зі словами TYPE і END_TYPE. До типу даних користувача відносяться перелічення, структури, масиви і масиви структур.

Щоб вставити робочі листки типу даних у проектне дерево, треба позначити теку Data Types (Типи даних). В панелі інструментів клацнути по іконі Add Objiect (Додати об'єкт) або в підменю Project (Проект) вибрати пункт Add Data Type (Додати тип даних).

У діалоговому вікні Insert (Вставка), що з'явитьсь ввести ім'я нового робочого листка типу даних і підтвердити діалог. Новий робочий листок типу даних буде вставлено у відповідне піддерево, рис. 1.4.

Щоб відкрити робочий листок типу даних з текстовим редактором, необхідно у дереві проекту подвійно клацнути по іконі Data Types Worksheet (Робочий листок типу даних).

Примітка. Якщо ікона Add Objiect (Додати об'єкт) інертна, захист з використанням пароля активізований. Тому треба ввести пароль використовуючи пункт меню Enter password (Ввести пароль) в підменю File (Файл).

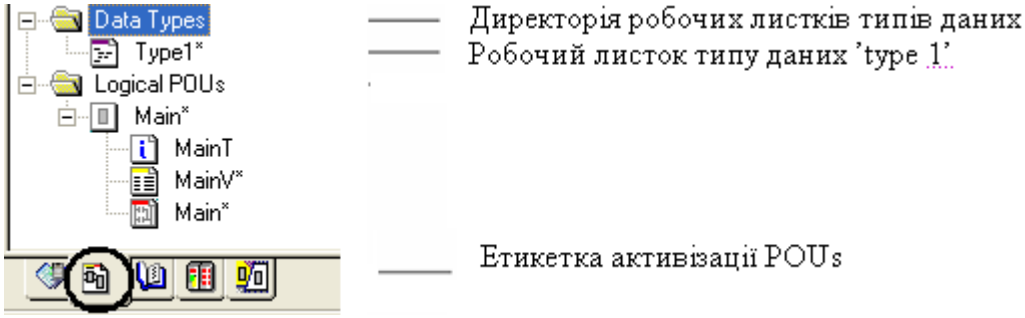


Рис. 1.4 Піддерево типів даних

Піддерево Data Type (Тип даних) можна оглядати, показуючи повне проектне дерево або тільки піддерева Data Type (Тип даних) і Logical POUs

(Логічні POUs). Для цього в меню проектного дерева, що розташоване внизу вікна, треба клацнути по етикетці POUs.

Тека POUs . Програми, функціональні блоки і функції можуть редагуватися у піддереві Logical POUs (Логічні POUs) проектного дерева. Кожне POU складається з декількох робочих листків, рис. 1.5.

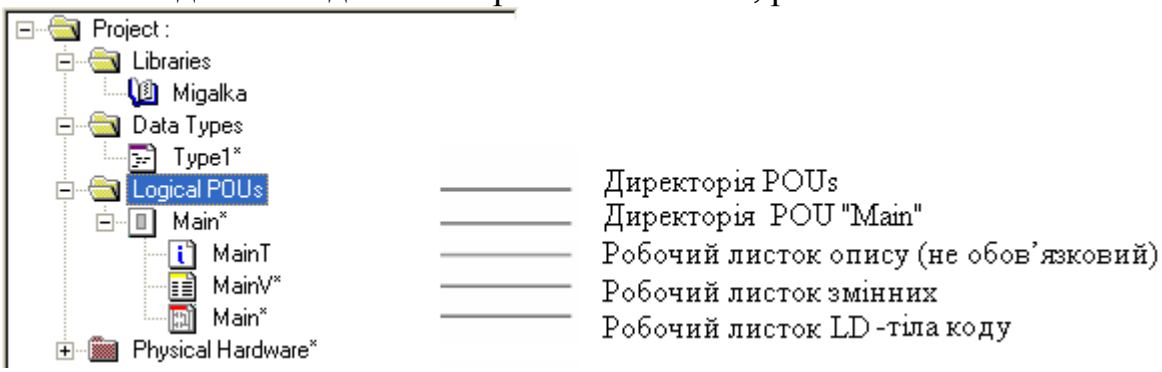


Рис. 1.5. Структура директорії POUs

Робочий листок опису створюється для документації і містить текст, введений користувачем, проте він не є обов'язковим. Робочий листок змінних містить декларовані локальні змінні. Робочий листок тіла коду містить фактичний код, що створювався однією з мов програмування. Ці робочі листки зображуються графічно іконами і мають назву, яка співпадає з назвою власного POU.

При програмуванні SFC – мовою робочі листки POUs доповнюються робочими листками дій та переходів. Якщо робочий листок коду відкритий, можна легко відкрити зв'язаний з ним робочий листок змінних, двічі клацнувши по відповідній іконі у піддереві Logical POUs (Логічні POUs) або вибравши пункт меню Open Variables Worksheet (Відкрити робочий листок змінних) у підменю View (Огляд), або клацаючи по іконі Variables Worksheet (Змінні робочого листка) в панелі інструментів.

Піддерево Logical POUs (Логічні POUs) містить усі POUs, які використовуються в проекті.

Можна оглядати піддерево Logical POUs (Логічні POUs), показуючи повне проектне дерево або тільки піддерева Data types (Тип даних) і Logical POUs (Логічні POUs). Для цього необхідно клацнути по етикетці POUs в меню проектного дерева, що розташоване внизу вікна.

Щоб вставити POUs у проектне дерево, треба позначити теку Logical POUs (Логічні POUs) і в панелі інструментів клацнути по одній з наступних ікон:

Add program (Додати програму), щоб вставити програму.

Add FB (Додати FB), щоб вставити функціональний блок.

Add Function (Додати функцію), щоб вставити функцію.

Альтернативно можна вибрати відповідний пункт меню у підменю Project > Add POU.

З'явиться діалог Insert (Вставка), в якому ввести ім'я нового POU, вибрати бажану мову, при необхідності ввести PLC і/або тип процесора і,

якщо потрібно, позначити перемикач Use Reserve (Використання резерву). Підтвердити діалог.

Use Reserve (Використання резерву) означає, який резерв пам'яті POU використовується для Patch POU (Вставка POU). Значення Use Reserve (Використання резерву) записано у діалозі Data Area (Область даних), який викликається натисканням кнопки Data Area (Область даних) у діалозі Resource setting (Установки ресурсу) вашого ПЛК.

Після цих дій новий POU є вставлений у проектне дерево. Він містить робочий листок тіла коду, робочий листок змінних і робочий листок опису. Нові робочі листки в проектному дереві позначені зірочкою, це означає, що вони ще не компільовані.

Примітка. Якщо право на вставлення POU у проектне дерево обмежене паролем користувача, з'явиться повідомлення про помилку. В даному випадку необхідно зареєструватися в проекті, використовуючи дійсний пароль проекту. Для цього потрібно відкрити підменю File (Файл) і вибрати пункт меню Enter password (Ввести пароль).

Теки елементів конфігурації

Елементи конфігурації у проектному дереві зображені графічно. Система програмування відображає структуру елементів конфігурації у піддереві Physical Hardware (Фізична апаратура).

Протягом створення робочого коду програм и функціональних блоків у піддереві Physical Hardware (Фізична апаратура), рис. 1.6, додаються їх екземпляри, а також редагуються окремі робочі листки глобальних змінних.

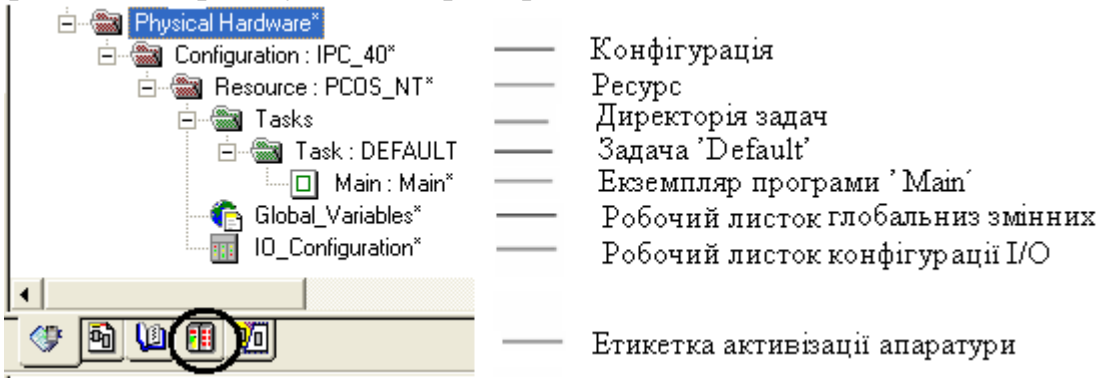


Рис. 1.6 Структура піддерева Physical Hardware

Як результат кожний екземпляр програми або функціонального блока має власний набір даних з безпосереднім доступом.

Екземпляри програми і функціонального блока корисні, якщо маємо, наприклад, дві ідентичні машини свердління, які керуються одним ПЛК через різні модулі вводу-виводу. Кожний екземпляр працює з даними його власної декларації глобальних змінних.

Щоб зробити видимим дерево екземпляра, необхідно клацнути по етикетці Instances (Екземпляри) в меню, що розташоване внизу вікна проектного дерева.

Завдання до лабораторної роботи

1. Зрозуміти методику створення нового проекту і засвоїти можливості редактора проектного дерева.

2. Створити новий проект, в якому оголосити власну бібліотеку, у проектному дереві додати робочі листки типу даних і POUs, захистити доступ до бібліотеки. Відкрити дерево екземпляра.

Контрольні запитання

1. З яких етапів складається створення нового проекту?
2. Як оголосити власну бібліотеку?
3. Яки типи даних розташовуються у теці Data Types проектного дерева?
4. Що треба зробити, щоб вставити робочі листки типу даних у проектне дерево?
5. Що таке POU і, з яких робочих листків він складається?
6. Як можна вставити POU у проектне дерево?
7. Як у проектному дереві додати робочі листки типу даних і POU?
8. Які елементи конфігурації містить проектне дерево?

Лабораторна робота №2
Програмування LD-мовою у Soflogic-системі KW MULTIPROG

Знайомство з редактором LD- мови

Для знайомства з редактором LD-мови необхідно за методикою, що приведена у лабораторній роботі №1, створити новий проект і, двічі клацнувши лівою клавішею миші по іконі Main* теки Logical POUs(Логічні POUs), відкрити робочий листок для створення тіла коду LD-мовою.

Створений новий проект Main з’явиться у вікні проектного дерева, рис.2.1, у піддеревах Logical POUs і Physical Hardware.

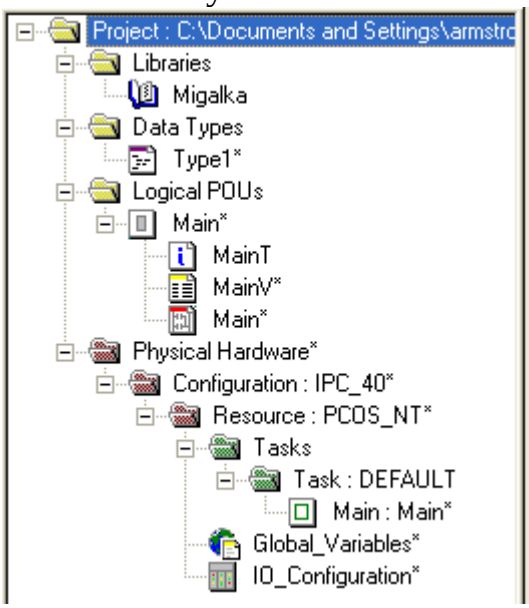


Рис. 2.1. Дерево проекту

Для створення LD- мережі необхідно:

- у зручному місці робочого листка тіла коду клацнути лівою клавішею миші, щоб позначити вихідну точку програмування;

- в панелі інструментів клацнути по іконі Contact network (Контактна мережа) і в робочому листку тіла коду з'явиться перша LD- мережа з одним контактом, одною котушкою і двома шинами живлення:

LD- мережу можна завжди розширити за рахунок додаткового контакту вставленого зліва або справа від існуючого контакту, а також котушки, яка вставляється тільки справа від контакту.

Щоб вставити в LD- мережу інший стандартний об'єкт:

- позначте в ній існуючий контакт або котушку;

- в панелі інструментів клацніть по іконі Contact right (Контакт праворуч) і контакт з'явиться праворуч або по іконі Cntact left (Контакт ліворуч), щоб контакт з'явився ліворуч;

Якщо в панелі інструментів клацнути по іконі Coil right (Котушка праворуч), праворуч з'явиться котушка.

Щоб вставити елементи LD-мови паралельно існуючим контактам і котушкам необхідно:

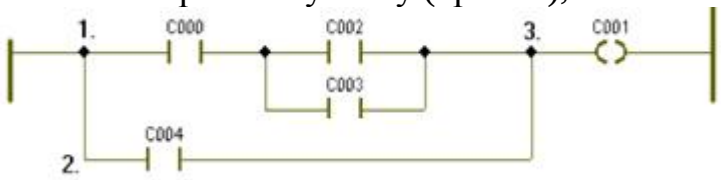
- позначити лівою клавішею миші контакт чи котушку;

- в панелі інструментів клацнути по іконі Parallel (З'єднати паралельно), щоб з'явився новий елемент нижче вибраного, або по іконі Add contact/coil above (Додати контакт/ котушку зверху), аби новий елемент з'явився вище вибраного.

Щоб вставити в LD-мережу паралельні гілки треба:

- в панелі інструментів клацнути лівою клавішею миші по іконі Branch (Вставити гілку), щоб активізувати режим редагування LD-гілок.

- клацнути лівою клавішею миші по лінії сполучення між двома об'єктами, де бажано почати паралельну гілку (крок 1);



- просуваючи мишу вперед або назад, нарисувати нову сполучну лінію у вільну зону (крок 2);

- клацнути лівою клавішею миші на рівні майбутнього розташування нового мовного об'єкта;

- перемістити курсор до бажаного кінця паралельної гілки (крок 3);

- клацнути лівою клавішею миші, щоб встановити зв'язок ліній.

Для вставлення у LD-мережу шин живлення, якщо їх немає:

- у LD- мережі позначити елемент, біля якого бажано зробити вставку;

- в панелі інструментів клацнути по іконі Left powerrail (Ліва шина живлення), щоб вставити ліву шину живлення, або по іконі Right powerrail (Права шина живлення), щоб вставити праву шину живлення.

Для видалення шини живлення позначте її лівою клавішею миші і натисніть “Delete” на клавіатурі комп’ютера.

Окремі LD- мережі можна з’єднати між собою. Ліві шини живлення сполучаються тільки з лівими, а праві - тільки з правими.

Для з’єднання LD- мереж:

- в панелі інструментів клацнути лівою клавішею по іконі Connect (З’єднати);
- позначити курсором першу шину живлення;
- клацнути послідовно по інших шинах живлення - і вони з’єднаються автоматично.

Для створення декількох паралельних мереж однакового розміру і вже з’єднаних одна з одною необхідно нижче лівої шини живлення попередньої мережі встановити мітку і в панелі інструментів клацнути по іконі Network (Мережа). Зробити це потрібно стільки разів, скільки додаткових мереж необхідно вставити.

Розмір LD-мережі можна змінити, для цього необхідно:

- у підменю Layout (Формат) вибрати пункт Contact Width (Ширина контакту). З’явиться діалог Contact Width (Ширина контакту);
- ввести нове значення ширини мережі і натиснути ОК. Після коригування розміру всі нові LD-мережі мають нову ширину.

Можна вирівняти LD-мережі, якщо їх праві шини живлення по різному розташовані. Редактор забезпечує дві можливості вирівнювання правих шин живлення:

1. Щоб вирівняти шини живлення за самою правою шиною треба:
   - установити курсор на саму праву шину живлення і клацнути лівою клавішею миші;
   - відкрити підменю Edit (Редагування), вибрати і активізувати пункт Stretch/ Compress > Arrange Powerrails (Розтягнення/ Стиснення > Видалення шин живлення). Усі шини живлення змістяться до рівня позначеної шини.
2. Щоб вирівняти шини живлення за міткою у робочому листку (вирівнювання здійснюється тільки зміщенням шин праворуч) треба:
   - клацнути лівою клавішею миші у необхідній точці робочого листка;
   - відкрити підменю Edit (Редагування), вибрати і активізувати пункт меню Stretch/ Compress > Arrange Powerrails (Розтягнення/стиснення > Видалення шин живлення). Усі шини змістяться на рівень встановленої мітки.

В LD- мережу можна викликати функції та функціональні блоки, які є елементами FBD- мови, тобто в тілі коду LD- програми змішуються елементи LD- і FBD- мови.

Функцію або функціональний блок можна вставити в будь-якому місці робочого листа, а потім з’єднати їх з іншими елементами. Можна також

вставити функцію або функціональний блок безпосередньо в існуючу LD-мережу.

Найзручніший шлях встановлення FBD функцій або функціональних блоків в LD- програму - це використання Edit Wizard (Майстра редагування).

Позначте лівою клавішею миші місце на LD- мережі, де бажаєте вставити функцію або функціональний блок;

- в панелі інструментів натисніть ікону Edit Wizard (Майстра редагування);

- у вікні, що з'явиться, виберіть зі списку Group потрібну функцію або функціональний блок і двічі клацніть по ньому лівою клавішею миші;

- у діалоговому вікні Variable properties (Властивості змінної) змініть типове ім'я функціонального блока на бажане і натисніть ОК, аби блок з'явився у позначеному місці.

В LD- мережі можна змінити властивості контактів і котушок.

Для цього необхідно:

- подвійно клацнути по контакту або котушці, що змінюється;

- у діалоговому вікні Contact/ Coil Properties (Властивості контакт/ котушка на сторінці Contact (Контакт) змінити тип контакту або котушки, використовуючи перемикачі Contact (Контакт) і Coil (Котушка), а також можливості вікна Type (Тип).

Для оголошення змінних контактів або котушок існує дві можливості:

а) використання змінних, які вже декларовані в робочому листку змінних;

б) використання змінних, які ще не декларовані в робочому листку змінних.

Щоб змінити типове ім'я контакту або котушки на вже існуюче треба:

- подвійно клацнути по контакту/котушці, для якого/якої бажано оголосити вже декларовану змінну. З'явиться діалогове вікно Contact/ Coil Properties (Властивості Контакт/ котушка).

На сторінці Contact (Контакт) список змінних у полі Name (Ім'я) містить усі локальні або глобальні змінні, які були раніше декларовані;

- у вікні списку змінних Name (Ім'я) клацнути по імені бажаної змінної, і позначене ім'я увійде у перший рядок текстового поля змінних;

- натиснути ОК, і вибране ім'я змінної з'явиться у робочому листку тіла коду, як ім'я позначеного контакту/ котушки.

Щоб оголосити нову змінну для контакту або котушки:

- подвійно клацнути по контакту/котушці, для якого/якої бажано декларувати нову змінну. З'явиться діалогове вікно Contact/Coil Properties (Властивості контакт/ котушка);

- у полі Name (Ім'я) ввести нове ім'я;

- вибрати для змінній контакт/котушка відповідні установки Usage(Вжити), Data Type(Тип Даних), I/O address(Адреса В/В);

- у полях Local Variable Groups (Групи локальних змінних) і Global Variable Groups (Групи глобальних змінних) вибрати групу змінних Default (За

замовчуванням), в яку бажано вставити опис нової змінної, і клацнути лівою клавішею миші.

Після вибору компетенції клацнути ОК. Змінна з'явиться у робочому листку тіла коду, як ім'я контакту/ котушки, а її декларація автоматично буде вставлена у відібрану групу робочого листка сітки змінних.

У разі глобальної змінної декларація вставляється у сітку локальних змінних, використовуючи ключове слово VAR_EXTERNAL і в таблицю глобальних змінних робочого листа, використовуючи VAR_GLOBAL.

Щоб вставити і декларувати нову змінну, яка не є контактом/ котушкою необхідно:

- у робочому листку тіла коду лівою клавішею миші позначити місце вставки нової змінної;

- в панелі інструментів клацнути по іконі Variable (Змінна) і у діалоговому вікніVariable Properties (Властивості змінної), що з'явиться, зробити все те, що робилося при оголошені змінної контакту або котушки.

Після закриття діалогового вікна змінна з'явиться у робочому листку тіла коду, а її декларація автоматично вставиться у сітку відповідних змінних.

Біля лівої шини живлення LD- мережі можна вставити коментарі. Для цього треба:

- подвійно клацнути по лівій шині живлення, з'явиться діалог Comment (Коментар);

- ввести необхідний коментар і натиснути ОК.

Розроблення коду програми

Розробити LD-мовою програму управління роботою двигуна, що вмикається триразовим натисканням стартової кнопки, а вимикається автоматично за 20с роботи.

Покрокове нагадаємо, як створити першу LD-мережу, як оголосити властивості об'єктів, що автоматично з'являться з першою LD-мережею, як вставити і з'єднати функціональний блок у робочому листку тіла LD- коду за допомогою Edit Wizard (Майстер редагування), як вставити і з'єднати контакт у робочому листку коду програми, як оголосити властивості окремих контактів і котушок, як вставити другу LD-мережу і надрукувати коментар мережі.

Вставлення першої LD- мережі:

- клацнути лівою клавішею миші у зручному місці робочого листка для установлення мітки першої вставки;

- в панелі інструментів клацнути по іконі Network (Мережа) і в установленому місці з'явиться LD- мережа 001 певної ширини з контактом і котушкою.

Для оголошення властивостей змінних, які автоматично з'являться з першою LD- мережею необхідно:

- двічі клацнути лівою клавішею миші по контакту C000, щоб оголосити змінну, яка має запустити двигун. У діалоговому вікні Contact/Coil Properties (Властивості Контакт/Котушка), що з'явиться, замінити типове ім'я змінної C000 на Motor_Start;

- у рядку Data Type (Тип даних) вибрати тип BOOL, а зі списку Usage (Вжити) – VAR.

Таким чином, змінна оголошується як локальна, а тому вона може використовуватися тільки в POU проекту.

Щоб призначити зовнішній змінній Motor_Start фізичну адресу симулятора модуля вводу/виводу ПЛК:

- у рядку I/O address (Адреса вводу/виводу) ввести %IX0.0, тут 0.0– позначає перший модуль вводу і його перший канал; I – фізичний вхід; X – однобітовий розмір змінної;

- зі списку Local Variable Groups: вибрати Default;

- клацнути ОК, підтверджуючи діалог Contact/Coil Properties (Властивості Контакт/Котушка). Після цього змінна є оголошеною і вставлена у відповідний список, а на робочому листку проекту замість C000 з'явиться ім'я Motor_Start.

Двічі клацніть по котушці C001, з'явиться діалогове вікно Contact/Coil Properties (Властивості Контакт/Котушка);

- у діалоговому вікні Contact/Coil Properties (Властивості Контакт/Котушка), що з'явиться, замінити типове ім'я змінної C001 на Motor;

- у рядку Data Type (Тип даних) вибрати тип BOOL, а зі списку Usage (Вжити) – VAR.

Для призначення зовнішній змінній Motor фізичної адреси симулятора модуля вводу/виводу I/0 PLC у рядку I/0 Address введіть %QX0.0,

тут Q – позначає фізичний вихід; 0.0 – перший вихідний модуль і його перший канал; X – однобітовий розмір змінної;

- оскільки двигун має працювати деякий час безперервно після його увімкнення, у рядку Type (Тип) виберіть котушку –(S)– (увімкнути вихід і зафіксувати);

- зі списку Local Variable Groups: вибрати Default і клацнути ОК, щоб завершити оголошення змінної і вставити її над котушкою і у відповідному списку змінних.

Вставлення в LD- мережу лічильника натисків стартової кнопки:

Оскільки двигун має увімкнутися після триразового натискання стартової кнопки вставимо в LD- мережу лічильник.

Клацніть по змінній Motor_Start, щоб вставити лічильник одразу за нею;

- в панелі інструментів лівою клавішею миші активізуйте ікону Edit Wizard (Майстер редагування) і у таблиці Group (Група), що з'явиться на екрані, двічі клацніть по блоку CTU;

- у діалоговому вікні Variable properties (Властивості змінної) змініть типове ім'я CTU-1 на Motor_Count і натисніть ОК. За позначеним контактом LD- мережі з'явиться лічильник;

- в панелі інструментів клацніть по іконі Edit Wizard (Майстер редагування), щоб сховати бібліотеку Майстра редагування.

Для визначення параметрів лічильника двічі клацніть по блакитній точці PV-входу, з'явиться діалогове вікно Variable properties (Властивості змінної);

- у рядку Name (Ім'я) введіть INT#3, щоб двигун (Motor) спрацював після триразового натискання стартової кнопки, тут INT – означає ціле число; # - означає константу; 3 – фактичне значення константи;

- клацніть ОК і на PV-вході лічильника з'явиться INT#3;

Двічі клацніть по зеленій точці CV-виходу лічильника, з'явиться діалогове вікно Variable properties (Властивості змінної);

- в рядку Name (Ім'я) введіть Pressed - це змінна, яка запам'ятовує поточне значення лічильника;

- у рядку Date type (Тип даних) виберіть INT, оскільки змінною є число, а у рядку Usage (Вжити) – VAR;

- клацніть ОК і у робочому листку поряд з CV-виходом з'явиться Pressed.

Вставлення контакту скидання лічильника:

- клацніть на функціональному блоці CTU по синій точці Reset-входу;

- в панелі інструментів клацніть по іконі Contact left (Контакт зліва) і на вході лічильника з'явиться контакт C002;

- активізуйте в панелі інструментів кнопку Connect (З'єднання);

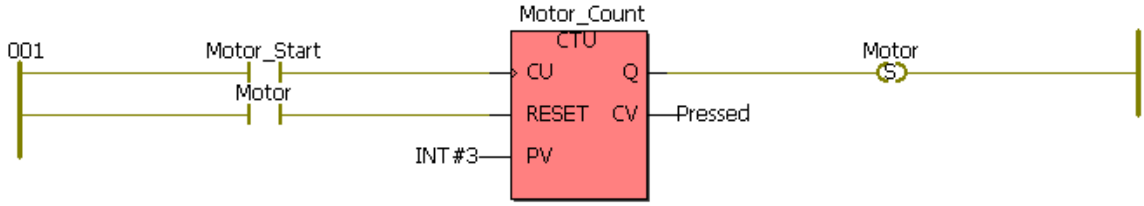
- клацніть по синій точці лінії контакту C002, перемістіть її курсором ліворуч до шини живлення 001 і повторно натисніть ліву клавішу миші, щоб з'єднати лінії між собою;

- в панелі інструментів клацніть по іконі Mark (Позначка), щоб деактивізувати кнопку Connect (З'єднання);

Оголошення властивостей контакту скидання лічильника:

- двічі клацніть по контакту C002 і у діалоговому вікні Contact/Coil Properties (Властивості Контакт/котушка) зі списку вже оголошених змінних, виберіть Motor.

Оскільки змінна була визначена раніше, немає потреби конкретизувати її властивості. У рядку Type (Тип) виберіть прямий контакт і клацніть ОК. На робочому листку з'явиться змінна Motor замість C002:



Вставлення другої LD- мережі

Клацніть лівою клавішею миші нижче першої LD- мережі, на робочому листку з'явиться мітка установлення нової мережі;

- в панелі інструментів клацніть по іконі Network (Мережа) і на екрані з'явиться друга LD- мережа;

- двічі клацніть по контакту C003 для оголошення його властивостей;

- у діалозі Contact/Coil Properties (Властивості Контакт/Котушка) із списку вже оголошених змінних виберіть Motor і клацніть ОК для зміни у LD-мережі імені контакту C003 на Motor .

Для вставлення таймера, що керує тривалістю роботи двигуна:

- клацніть лівою клавішею миші по контакту Motor;

- в панелі інструментів натисніть на ікону Edit Wizard (Майстер редагування), щоб на екрані з'явилося вікно Group (Група);

- у вікні Group (Група) виберіть Function blocks (Функціональні боки) і двічі клацніть лівою клавішею миші на таймері TON;

- у рядку Name (Ім'я) діалогова вікна Variable properties (Властивості змінної) введіть M_Time і натисніть кнопку ОК. Таймер з'явиться у LD- мережі за контактом Motor;

- клацніть по іконі Edit Wizard (Майстер редагування), щоб приховати вікно Group (Група);

- для визначення часу роботи таймера необхідно лівою клавішею миші двічі клацнути по блакитній точці його PT-входу;

- у діалоговому вікні Variable properties (Властивості замінної), що з'явиться, у рядку Name (Ім'я) ввести часову константу T#20s і натиснути ОК.

Тут T – позначає префікс часу; # - означає константу; 20s – час роботи двигуна (20с);

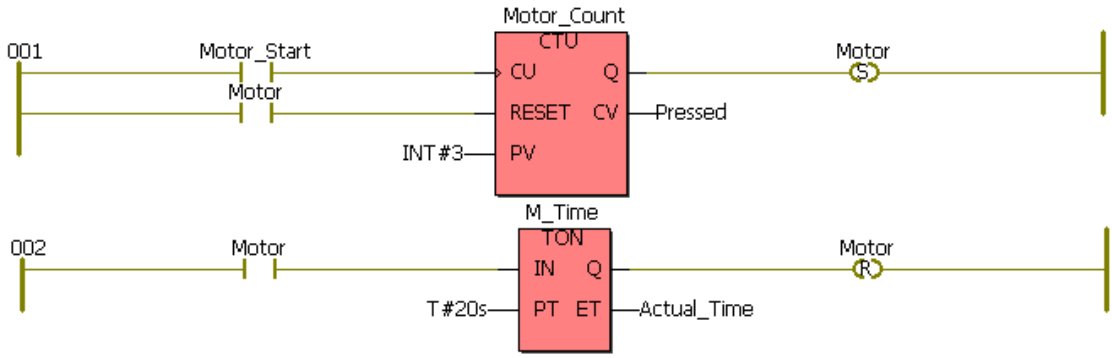
Для визначення фактичного часу роботи таймера двічі клацніть по зеленій точці ET-виходу таймера;

- у діалоговому вікні Variable properties (Властивості змінної), що з'явиться, у рядку Name (Ім'я) введіть Actual_Time (Фактичний час), як ім'я локальної змінної;

- зі списку Data_Type (Тип даних) виберіть Time (Час), клацніть ОК і на ET-виході таймера з'явиться Actual_Time;

- двічі клацніть по котушці C004, відкриється діалогове вікно Contact/Coil Properties (Властивості Контакт/Котушка);

- для зупинення двигуна виберіть курсором зі списку змінних ім'я Motor, у вікні Type (Тип) - котушку –(R)– (вимкнути вихід і зафіксувати) і клацніть ОК.



Присвоєння імені розробленої мережі:

- двічі клацнути по лівій шині живлення першої LD- мережі, з'явиться діалогове вікно Comment (Коментар);

- у діалоговому вікні Comment (Коментар) надрукувати Circuit (Схема) і клацніть Font>> (Шрифт);
- для зміни властивостей шрифту у вікні, що з'явиться, вибрати блакитний колір і розмір 20;
- клацнути ОК у вікні Font (Шрифт) і у вікні Comment (Коментар). Над розробленою мережею з'явиться ім'я Circuit.

Компілювання проекту

Тепер, коли редагування закінчено, необхідно зробити компілювання проекту. Протягом компілювання вміст робочих листків перетворюється у спеціальний код, який може бути виконаний ПЛК.

Оскільки у цьому прикладі немає ПЛК, треба активізувати стимулятор. Для цього:

- у дереві проекту клацнути правою клавішею миші по теці Resource (Ресурс) і в контекстному меню Settings (Установки) у вікні Resource settings for IRC_40 (Установки ресурсу для IPC-0), що з'явиться, активізувати Simulation 1 (Симуляція 1), якщо це необхідно і закрити діалог, клацнувши ОК;
- в панелі інструментів клацнути по іконі Make (Утворювати);
- у вікні Build (Конструкція) внизу екрана відобразяться помилки (Error) і попередження (Warning), якщо будуть знайдені протягом компіляції.

Помилки синтаксичного або структурного характеру перешкоджають завершенню процесу компіляції.

Попередження вказують на потенційні проблеми, подібно змінній, якої немає, а вона використовується. Попередження не перешкоджають завершенню процесу компілювання. Попередженнями можна нехтувати, а помилки потрібно виправити.

Для виводу на екран списку знайдених помилок у вікні Build (Конструкція) клацнути по Error (Помилки), а для виводу на екран списку попереджень у вікні Build (Конструкція) клацнути по Warning (Попередження).

Щоб виправити усі помилки:

- двічі клацнути по вказаній помилці або попередженню, відкриється відповідний робочий листок;
- виправити усі помилки і попередження і повторно компілювати проект, клацнувши в панелі інструментів по іконі Make (Утворювати);

Тільки після цього можна завантажувати програму до симулятора.

Завантаження проекту до симулятора

Зв'язок з ПЛК або симулятором здійснюється завдяки діалогу Resource (Ресурс), який керує контролером. Для цього:

- клацнути по іконі Project Control Dialog (Діалогове управління проектом), з'явиться діалогове вікно Resource (Ресурс) для управління ПЛК або симулятором;
- натиснути кнопку Download (Завантаження), з'явиться відповідне діалогове вікно;

- продублювати у лівій частині вікна натискання на кнопку Download і проект завантажиться у пам’ять симулятора;
- успішний процес завантаження проекту супроводжується бігом синьої стрічки внизу екрана.

Налагодження проекту

Робочі листки проекту можуть перемикатися з режиму редагування на режим налагодження і навпаки, за допомогою ікони Debug on/off (Налагодження В/В) в панелі інструментів. Коли режим налагодження активізовано, стан і поточні значення змінних на робочому листку проекту показані різними кольорами: блакитний – хибний, червоний – істинний.

Режим налагодження використовується для знаходження помилок програмування і переконання, що програма ПЛК працює правильно. Для запуску програми у діалозі управління Resource (Ресурс) натиснути кнопку Cold (Холодний) .

- Для виводу на екран симулятора модулів I/O (Вводу/виводу) клацнути по Demoio - Driver, що видно внизу монітора;
- розмістити симулятор таким чином, щоб проект на робочому листку не був прихований;
- тричі клацнути по зеленому віртуальному світлодіоду з адресою IX0.0 і спостерігати реакцію схеми в робочому листку проекту.

Motor (Двигун) почне працювати після того, як з’явиться цифра три біля Pressed, тому що поточне значення CV- виходу на блоці Motor_Count досягає установленого значення на PV- вході того ж блока.

Коли котушка Motor у мережі 001 вмикає двигун, вмикається контакт Motor у мережі 002 і таймер M_Time починає відлік часу протягом 20с. Тривалість руху двигуна продовжується доки час, що фіксує Actual_Time (Фактичний час) біля ET-виходу таймера, не співпаде з уставкою біля PT-входу (T#20s). По завершенню роботи таймера котушка –(R)– у мережі 002 вимикає двигун (Motor), що у мережі 001.

Оперативне редагування

Оперативне редагування можливе без зупинки програми, яка реалізується на ПЛК або симуляторі. Цей режим викликається, клацаючи в панелі інструментів по іконі Patch POU (Вставка POU) .

Коли використовується режим Patch POU (Вставка POU), зміни, що зроблені в коді програми автоматично завантажуються до ПЛК або симулятора. Протягом всього часу роботи в режимі Patch POU (Вставка POU) виконання коду програми не припиняється.

Як приклад роботи в режимі Patch POU (Вставка POU), вставимо у програму змінну Emergency_Stop (Непередбачена зупинка). Активізація Emergency_Stop (Непередбачена зупинка) негайно зупинить двигун.

- Клацніть по іконі Debug on/off (Налагодження В/В), щоб перевести робочий листок коду програми в режим редагування. Проте, ресурс так само, як і реальний контролер, працює;

- установіть курсор на робочому листку коду програми нижче шини живлення OO2 і клацніть лівою клавішею;

- в панелі інструментів клацніть по іконі Contact Network (Мережа контактів), і на робочому листку з'явиться нова LD- мережа 003;

- двічі клацніть по контакту C005, відкриється діалог Contact/Coil Properties (Властивості Контакт/Котушка). У рядку Name (Ім'я) замініть типове ім'я C005 на Emergency_Stop (Непередбачена зупинка);

- клацніть OK, автоматично відкриється сторінка діалогу Common (Загальний);

- з вікна списку Usage (Вжити) виберіть VAR_EXTERNAL, декларуючи таким чином, Emergency_Stop (Непередбачена зупинка), як глобальну змінну.

Для непередбаченої зупинки двигуна використаємо другий канал симулятора I/O:

- у рядку I/O Address (Адреса В/В) введіть % IXO.1 Клацніть OK, у мережі 003 з'явиться змінна Emergency_Stop замість C005;

- двічі клацніть по контакту C006, з'явиться діалог Contact/Coil Properties (Властивості Контакт/Котушка);

- з вікна списку Type (Тип) виберіть –(R)–, а з вікна списку Variable (Змінна) – Motor. Клацніть OK, і на робочому листку з'явиться третя відкоригована LD- мережа.

Тепер, коли код програми змінено, використаємо режим Patch POU (Вставка POU), щоб компілювати змінні і завантажити їх до симулятора без його зупинки.

В панелі інструментів клацніть по іконі Patch POU (Вставка POU), змінений код проекту компілюється і завантажиться у симулятор I/O.

Після того, як режим Patch POU (Вставка POU) успішно завершиться, робочий листок буде автоматично переведений до оперативного режиму.

Клацніть по Demoio-Driver, щоб відкрити симулятор I/O;

- тричі подвійним клацанням по нульовому віртуальному світлодіоду нульового модуля I/O змінити стан біту;

- використовуючи новий контакт Emergency_Stop (Непередбачена зупинка), негайно зупиніть двигун Motor, клацаючи по світлодіоду 1-го нульового модуля вводу.

Протягом оперативного редагування можна використовувати можливості вікна спостереження.

Вікно спостереження змінних – це могутній інструмент, що дозволяє легко вставити різні змінні у список і спостерігати динаміку їх поведінки. Як тільки змінна додана у вікно спостереження, не потрібно відкривати відповідний робочий листок для контролю за нею. Таким чином, можна зосередитися лише на змінних, які потрібні для аналізу роботи програми.

- Клацніть по іконі Debug on/off (Налагодження В/В), щоб перевести робочий листок в оперативний режим;

- в панелі інструментів клацніть по іконі Watch Window (Вікно спостереження), унизу екрана з'явиться вікно спостереження;

- установіть курсор на змінну Motor_Start і клацніть правою клавішею миші;
- у контекстному меню, що з'явиться, виберіть Add to Watch Window (Додати до вікна спостереження) і клацніть лівою клавішею миші, змінна Motor_Start буде внесена до списку;
- продублюйте такі самі дії зі змінними Pressed (Натиснути) і Actual_Time (Фактичний час). У вікні спостереження з'явиться список з трьох змінних;
- натисніть Demoio-Driver для виклику симулятора I/O;
- тричі подвійно клацніть по нульовому світлодіоду нульового модуля I/O, щоб запустити програму, і спостерігайте стан змінних одночасно на робочому листку і у вікні спостереження.

Завдання для самостійної роботи

1. Розробити програму керування роботою двох двигунів, які після запуску одного з двигунів безперервно по черзі вмикаються і вимикаються. Тривалість роботи двигунів відповідно 5 і 10 секунд. Початковий запуск першого двигуна здійснюється одноразовим натисканням пускової кнопки, а початковий запуск другого двигуна – дворазовим. При цьому двигуни можна у будь-яку мить зупинити, а кожне вмикання двигунів підраховується лічильником. Коли кількість вмикань двигунів досягає десяти, їх робота автоматично зупиняється.

Налагодження програми здійснити, використовуючи вікна спостереження змінних і перехресне довідкове вікно.

2. Розробити програму нескінченного миготіння лампи, в якій довжина імпульсу у двічі більше паузи.

Контрольні запитання

1. Як розробляється код програми?
2. У чому різниця між прямим і інвертованим контактами?
3. Коли асоційована змінна 'Set' котушки приймає істинне значення?
4. Що трапляється з асоційованою змінною "Reset" котушки, коли стан лівого сполучення дорівнює TRUE?
5. Як оголошуються змінні при створенні додатку користувача?
6. Як оголосити функціональний блок у LD-програмі?
7. Як здійснюється компіляція і завантаження проекту в симулятор?
8. Як відбувається налагодження роботи програми?

Лабораторна робота №3
Програмування FBD-мовою у Soflogic-системі KW MULTIPROG

Знайомство з редактором FBD-мови

Графічний редактор FBD-мови має багато можливостей, які полегшують створення тіла коду програми, функціонального блока або функції.

Щоб вставити функцію або функціональний блок у робочий листок тіла коду за допомогою Edit Wizard (Майстер редагування) необхідно:

– у робочому листку тіла коду визначити положення нової функції або функціонального блока і клацнути лівою клавішею миші;

- в панелі інструментів натиснути ікону Edit Wizard (Майстер редагування), щоб Майстер редагування був видимий;

– у вікні Group (Група) Майстра редагування відкрити список функцій або функціональних блоків і подвійно клацнути по бажаному об'єкту.

Коли вставляється функція, то вона миттєво з'являється у робочому полі після подвійного натиснення лівої клавіші миші.

Коли вставляється функціональний блок, з'являється діалогове вікно Variables Properties (Властивості Змінних), де у полі Name (Ім'я) сторінки Variables (Змінні) пропонується типове ім'я екземпляра блока (наприклад, для CTU пропонується ім'я CTU_n, де n – порядковий номер екземпляра вибраного блока).

Щоб присвоїти унікальне ім'я новому функціональному блоку, введіть його у перший рядок поля Name (Ім'я) замість запропонованого або виберіть зі списку вже декларованих змінних, що приведені нижче у полі Name (Ім'я).

Може виникнути необхідність заміни вставлених функцій або функціональних блоків на інші. Для цього необхідно:

– у робочому полі тіла коду клацнути по функції або функціональному блоку, який потрібно замінити. Позначений об'єкт змінює свій колір;

– використовуючи Майстра редагування вибрати бажану функцію або функціональний блок з вікна Group(Група);

– подвійно клацнути по вибраній функції або функціональному блоку і зміна відбудеться.

Властивості функцій та функціональних блоків при необхідності можна змінити. Для цього:

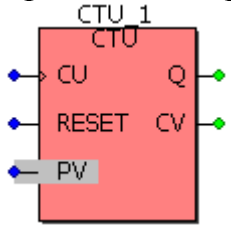
– у робочому листі FBD- коду правою клавішею миші клацнути по функції або функціональному блоку, щоб відкрити контекстне меню;

– вибрати пункт Object Properties (Властивості об'єкта) і клацнути лівою клавішею, з'явиться діалог FB/FU Properties (Властивості FB/FU). Коли змінюються властивості функції, з'являється тільки діалогова таблиця FB/FU, яка показує формальні параметри для зміни. У разі функціонального блока діалогове вікно FB/FU Properties (Властивості FB/FU) має чотири сторінки, в яких потрібно змінити бажані властивості і закрити діалогове вікно. Нова декларація екземпляра функціонального блока автоматично вставляється у групу локальних змінних.

Оголошення змінних може бути зроблено в будь-якому місці робочого листка тіла коду або після безпосередньої прив'язки їх до входу/виходу (формальному параметру) функції або функціонального блока.

Якщо є необхідність оголосити змінну не прив'язану до функції або функціонального блока, клацніть лівою клавішею миші у робочому листку тіла коду, щоб позначити місце вставки змінної.

Якщо є необхідність оголосити змінну вже зв'язану з функцією або функціональним блоком, позначте формальний параметр об'єкта:



– в панелі інструментів клацніть по іконі Variable (Змінна), з'явиться діалогове вікно Variable Properties (Властивості змінної);

– вікно списку змінних рядка Name (Ім'я) містить усі локальні або глобальні змінні, які були задекларовані раніше;

– клацніть по бажаній змінній у вікні списку змінних Name (Ім'я) і позначене ім'я увійде у перший рядок цього;

– закрийте діалогове вікно і змінна вставиться у позначеному місці робочого листка тіла коду.

Щоб оголосити і вставити нову змінну:

– в панелі інструментів клацніть по іконі Variable (Змінна), з'явиться діалогове вікно Variable Properties (Властивості змінної);

– заповніть необхідні рядки для нової змінної – Name (Ім'я), Usage(Вжити), Data Type(Тип Даних), I/O address(Адреса В/В);

– у сторінках Definition scope (Визначення компетенції) визначтесь оголошується локальна чи глобальна змінна, активізуючи Default (За замовчуванням).

Після вибору компетенції клацніть ОК і змінна вставиться у робочий листок тіла коду, а її декларація автоматично – у вибрану групу робочого листка сітки змінних. У разі глобальної змінної декларація вставляється у сітку локальних змінних, використовуючи ключове слово VAR_EXTERNAL, і у робочий листок сітки глобальних змінних, використовуючи VAR_GLOBAL.

Можна зробити заміну змінних, якщо з входом або виходом функціонального блока з'єднали помилкову змінну. Для цього:

Подвійно клацнути по змінній, яка замінюється. З'явиться діалог Variable Properties (Властивості змінних).

Для заміни вже декларованою змінною, в діалоговій сторінці Variables (Змінні) вибрати ім'я зі списку змінних і закрийте діалогове вікно.

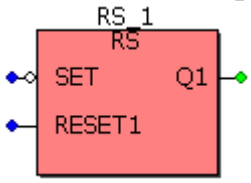
Для заміни поточної змінної новою введіть нове ім'я, зробіть необхідні оголошеня, визначтесь з компетенцією у сторінках Definition scope (Визначення компетенції), а потім закрийте діалогове вікно.

Використовуючи графічний редактор, легко зробити заперечення входів, виходів функцій та функціональних блоків, а також відмінити їх.

Для створення заперечення можна використати діалог FB/FU Properties (Властивості FB/FU) або виконати процедуру, яка описана нижче.

Клацніть лівою клавішею миші по входу або виходу, в якому потрібно зробити заперечення;

– в панелі інструментів клацніть по іконі Toggle negation of FP (Перемикати заперечення FP) і на позначеному вході або виході (формальному параметрі) з'явиться маленьке коло - символ заперечення:



Щоб відмінити заперечення, необхідно знову зробити такі самі кроки для того ж формального параметру.

При використанні діалогу FB/FU Properties (Властивості FB/FU), який викликається подвійним клацанням по функції або функціональному блоці, необхідно в ньому лише активізувати відповідну альтернативу вибраного формального параметра (Formal Parameters) і натиснути ОК.

Можна додавати входи функціям, але тільки тим, які працюють із декількома входами. При цьому дублюється тільки останній вхід функції.

Для дублювання входу функції можна використати діалог FB/FU Properties (Властивості FB/FU) або виконати процедуру, яка описана нижче.

– Клацніть по останньому входу функції, який треба дублювати, і він змінить колір;

– в панелі інструментів клацніть по іконі Duplicate FP (Подвійний FP) щоб встановити новий вхід.

При використанні діалогу FB/FU Properties (Властивості FB/FU), необхідно подвійним клацанням лівою клавішею миші по функції викликати діалогове вікно і позначити в ньому ім'я останнього входу функції, а потім послідовно натиснути кнопки Duplicate (Дублювати) та ОК.

Щоб видалити входи, які раніше були дубльовані, необхідно позначити їх і натиснути клавішу Delete(Видалити) на клавіатурі ЕОМ або позначити їх імена у діалоговому вікні FB/FU Properties (Властивості FB/FU) і послідовно натиснути кнопки Delete(Видалити) та ОК.

Розроблення проекту користувача

Створити FBD-мовою програму керування роботою двигуна згідно з алгоритмом наведеним у лабораторній роботі №2.

Для цього відкрийте діалог New Project (Новий проект), використовуючи пункт меню File (Файл). Двічі клацніть по Project Wizard (Майстер проекту) і пройдіть увесь шлях створення проекту аналогічно попередньому варіанту, тобто призначте ім'я проекту та його POU, виберіть FBD-мову програмування, бажані імена і типи конфігурації, ресурсу і задачі.

Щоб установити на робочому полі проекту функціональний блок лічильника для підрахунку кількості натисків на кнопку старту двигуна необхідно:

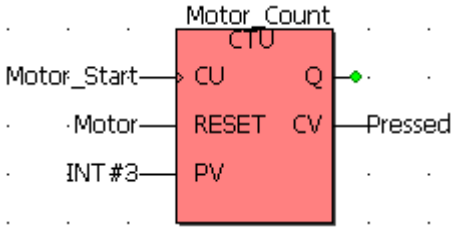
- вибрати на робочому полі місце установки першого функціонального блока і клацнути лівою клавішею миші;

- в панелі інструментів активізувати ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, вибрати функціональний блок CTU;

- подвійно клацнути по ньому і у діалоговому вікні Variable properties (Властивості змінної) змінити типове ім'я CTU-1 на Motor-Count;

- клацнути ОК, діалогове вікно закриється, а на робочому листку проекту залишиться функціональний блок Motor_Count.

Оголошення змінних блока Motor_Count здійснити, користуючись придбаним досвідом у попередній роботі та наведеною копією лічильника:



Для з'єднання функціонального блока лічильника Motor_Count з функціональним блоком домінантного перемикача RS і оголошення його вихідної змінної:

- лівою клавішею миші активізуйте Q-вихід на функціональному блоці Motor_Count;

- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, виберіть функціональний блок перемикача RS, який має домінанту вимикача;

- двічі клацніть по ньому і у діалоговому вікні Variable properties (Властивості змінної) у рядку Name з'явиться ім'я екземпляра блока RS_1;

- клацніть ОК, діалогове вікно закриється, а на робочому листку проекту з'явиться екземпляр функціонального блока RS_1 сполучений з функціональним блоком Motor_Count;

- лівою клавішею миші активізуйте Q-вихід на функціональному блоці RS_1 і оголосіть вихідну змінну Motor зі списку вже оголошених змінних проекту.

Оголошення змінних, які мають зупинити роботу двигуна:

Оскільки двигун має зупинятися після закінчення установленого часу роботи, а також примусово у будь-який момент, використаємо з бібліотеки MULTIPROG логічний оператор OR.

- Виберіть на робочому полі проекту місце розташування логічного оператора OR і клацніть лівою клавішею миші;

- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, виберіть логічний оператор OR;

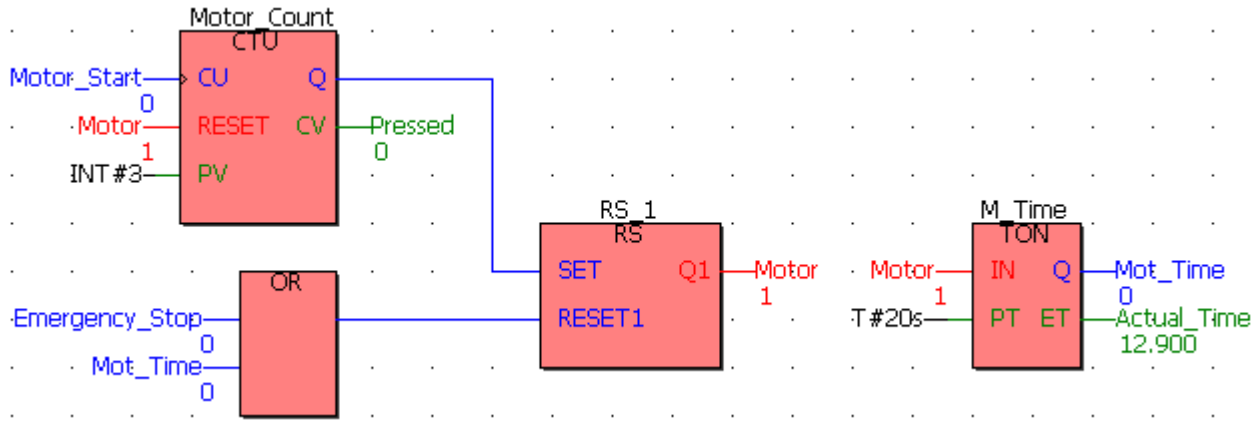
- двічі клацніть по ньому і на робочому полі проекту з'явиться OR;

- активізуйте перший вхід логічного оператора і оголосіть змінну Emergency_Stop(Непередбачена зупинка), яка забезпечить непередбачену зупинку двигуна, з адресою вхідного каналу модуля вводу %IX0.1;

- активізуйте другий вхід логічного оператора і оголосіть змінну Mot_Time, яка здійснить зупинку двигуна за 20 секунд роботи.

Визначення часу роботи двигуна здійснити за допомогою таймера M_Time із затримкою вмикання, у якого оголошені на IN-вхіді змінна Motor, на PT-вході константа часу 20с, на ET-виході змінна Actual_Time і на Q-виході змінна Mot_Time.

Для компіляції проекту, в панелі інструментів натисніть ікону Make (Створювати). Якщо помилки є, відкоригуйте проект, якщо немає - завантажте його, використовуючи відомий шлях: Project Control Dialog → Download→Download і перевірте роботу програми на симуляторі:



Завдання для самостійної роботи

1. Розробити програму керування роботою двох двигунів, які після запуску одного з двигунів безперервно по черзі вмикаються і вимикаються. Тривалість роботи двигунів відповідно 5 і 10 секунд. Початковий запуск першого двигуна здійснюється одноразовим натисканням пускової кнопки, а початковий запуск іншого двигуна – дворазовим. При цьому двигуни можна у будь-яку мить зупинити, а кожне вмикання і вимикання двигунів підраховується лічильником. Коли загальна кількість вмикань двигунів досягає десяти, їх робота автоматично зупиняється.

2. Розробити програму, яка забезпечує послідовне вмикання з інтервалом 2с чотирьох світлодіодів, а потім послідовне вимикання їх у зворотному напрямі з таким самим інтервалом. Інтервал часу при вмиканні програмувати за допомогою таймера TON, а при вимиканні – таймера TOF.

Контрольні запитання

1. Як здійснюється програмування FBD-мовою?
2. Як оголошуються змінні у функціональному блоці?
3. Як вставити функціональний блок у робочий листок тіла FBD-коду?
4. Як змінити властивості функції або блока?
5. Як додавати входи функціям і як зробити заперечення входів?
6. Як здійснити перекомпіляцію проекту?
7. Як перевірити роботу програми?
8. Як з’єднати функціональні блоки між собою?

Лабораторна робота №4
Програмування IL-мовою у Soflogic-системі KW MULTIPROG

Знайомство з редактором IL *–мови*

Тіло коду IL-мовою редагується текстовим редактором, який друкує інструкції або вставляє їх у робочий листок за допомогою Edit Wizard (Майстер редагування). При цьому елементи синтаксису IL-мови зображуються різними кольорами: оператори – блакитні; змінні й імена зразків – чорні; коментарі – зелені.

Для того, щоб редагувати тіло коду IL-мовою, необхідно у проектному дереві відкрити робочий листок, подвійно клацаючи по відповідній іконі теки Logical POUs(Логічні POUs).

За допомогою Майстра редагування інструкції вставляються у робочий листок тіла коду із вже завершеною структурою і користувач її просто заповнює змінними.

Якщо Edit Wizard (Майстер редагування) не активізований, треба натиснути на клавіатурі <SHIFT>+<F2> або <Alt>+<3>, або в панелі інструментів клацнути по іконі Edit Wizard.

Щоб вставити інструкцію:

– відкрийте робочий листок тіла коду, курсором визначте положення нової інструкції на екрані і клацніть лівою клавішею миші;

– у відкритому вікні Майстра редагування зі списку Group (Група), виберіть Operators (Оператори) і клацніть лівою клавішею. Майстер редагування покаже доступні оператори;

– виберіть необхідний оператор і двічі клацніть по ньому, щоб вставити у позначене місце. Деякі оператори вставляються окремо, а деякі - разом з іншими, відповідно до синтаксису IL-мови. Наприклад, при вставлені оператора ADD, в робочому полі з’являється така мовна структура:

```
LD   (* IN1 as ANY_NUM *)
ADD  (* IN2 as ANY_NUM *)
ST   (* Result as ANY_NUM *)
```

Щоб декларувати фактичну змінну біля оператора, позначте курсором змінну, що з’явиться з мовною структурою за замовчуванням і натисніть<F5> або в панелі інструментів клацніть по іконі Variable (Змінна). Відкриється діалогове вікно Variables (Змінні).

Для того, щоб вставити в мовну структуру змінну, яка вже декларована:

– на сторінці Variables (Змінні) діалогового вікна Variables (Змінні) у списку Name (Ім’я) вже декларованих змінних позначте бажане ім’я і клацніть лівою клавішею миші;

– позначене ім’я увійде у перший рядок списку Name (Ім’я);

– у рядку Local Variable Groups (Група локальній змінній) активізуйте Default і закрийте діалог. Змінна опиниться у позначеному місці робочого листка тіла коду.

Щоб декларувати нову змінну:

– надрукуйте нову змінну у бажаній позиції коду; наприклад, LD T_value;

– позначте курсором нове ім’я змінної і в панелі інструментів клацніть по іконі Variable (Змінна);

– з’явиться діалогове вікно Variables (Змінні) з надрукованим іменем T_value у першому рядку поля Name (Ім’я);

– виберіть відповідні установки для змінної у рядках Usage (Вжити), Data Type (Тип Даної), а також в інших, якщо потрібно,

– а у рядку Local Variable Groups (Група локальній змінній) активізуйте Default (За замовчуванням) і закрийте діалогове вікно.

Змінна вставиться у робочий листок тіла коду, а її декларація автоматично увійде у робочий листок сітки відповідних змінних.

Можна замінити змінну на таку, яка вже оголошена в робочому листку сітки змінних. Можна також замінити ім’я змінної на нове і таким чином декларувати при заміні нову змінну.

Щоб замінити змінну:

– позначте змінну, яка змінюється, курсором і натисніть <F5>. З’явиться діалогове вікно Variables (Змінні);

– для заміни поточної змінної на вже декларовану змінну виберіть у діалоговій сторінці Variables (Змінні) зі списку змінних бажане ім’я і далі продовжуйте, як при вставленні вже оголошеної змінної.

Для заміни поточної змінної новою, у діалоговій сторінці Variables (Змінні) введіть нове ім’я, а потім продовжуйте, як при декларуванні нової змінної.

Для зміни властивостей поточної змінної відкрийте діалогову сторінку Variables (Змінні), зробіть необхідні заміни і закрийте діалогове вікно.

Щоб в текстовому редакторі викликати функцію, використовується її ім’я у якості оператора та відповідні параметри, як це показано в наступному прикладі:

```
LD Inpar1
Funktion name   par2,par3
ST  var1
```

Перший декларований вхідний параметр «inpar1» завантажується у попередній рядок викликаної функції. Усі інші вхідні параметри «par2, par3» записуються через кому в другому рядку, як операнди. Результат зберігається змінною «var1», як це показано в останньому рядку прикладу.

Більш зручніше і надійніше редагування викликаної функції здійснювати за допомогою Майстра редагування (Edit Wizard). Якщо Майстер редагування невидимий на екрані, в панелі інструментів клацніть по іконі Edit Wizard (Майстер редагування);

– виберіть в робочому листку тіла коду позицію, де нова функція має бути вставлена, і клацніть лівою клавішею миші;

– відкрийте список Group (Група) у вікні Майстра редагування і виберіть курсором Functions (Функції). З’явиться список доступних функцій;

– виберіть бажану функцію і подвійно клацніть по ній лівою клавішею миші. Функція автоматично з'явиться у зазначеному місці робочого поля;

– замініть зелені коментарі (оточені круглими дужками і зірочками) необхідними елементами.

Наступний приклад показує вже редаговану функцію GE, яка вставлена в робочий листок, використовуючи Майстра редагування.

```
LD   (* IN1 as ELEMENTARY *)
GE   (* IN2 as ELEMENTARY *)
ST   (* Result as BOOL *)
```

Функціональні блоки викликаються в IL-редакторі за допомогою оператора CAL й імені зразка функціонального блока, як операнда.

Якщо Майстер редагування невидимий на екрані, в панелі інструментів клацніть по іконі Edit Wizard (Майстер редагування).

– Виберіть позицію в тілі коду, де буде вставлено новий функціональний блок. Клацніть лівою клавішею миші;

– відкрийте список Group (Група) у вікні Майстра редагування (Edit Wizard) і виберіть групу Function blooks (Функціональні блоки). Майстер редагування покаже доступні функціональні блоки;

– подвійно клацніть по бажаному функціональному блоку. З'явиться діалогове вікно Variables (Змінні), де у полі Name (Ім'я) за замовчуванням пропонується ім'я екземпляра блока (наприклад, для функціонального блока CTU пропонується ім'я „CTU_n" де n – порядковий номер цього імені екземпляра).

Щоб призначити екземпляру функціонального блока бажане ім'я:

– введіть у перший рядок діалогової сторінки Variables (Змінні) нове ім'я зразка або виберіть ім'я з текстового поля Name (Ім'я);

– натисніть ОК, і функціональний блок вставиться у позицію, що вказана курсором, а його декларація автоматично увійде у групу локальних змінних;

– змініть підказки, що оточені круглими дужками, на імена вхідних і вихідних змінних, необхідними.

Наступний приклад показує вже редагований CTU- функціональний блок, викликаний за допомогою Майстра редагування.

```
LD   (* BOOL *)
ST   CTU_1.CU
LD   (* BOOL *)
ST   CTU_1.RESET
LD   (* INT *)
ST   CTU_1.PV
CAL  CTU_1
LD   CTU_1.Q
ST   (* BOOL *)
LD   CTU_1.CV
ST   (* INT *)
```

Вже редагований функціональний блок CTU з іменем CTU_1

Зелені коментарі займають місця, які мають бути переписані фактичними значеннями й іменами

В IL робочих листках можуть використовуватися стрибки за допомогою оператора JMP, а також модифікаторів „C” або „CN” і мітки.

Щоб редагувати стрибки:

– встановіть курсор у бажане місце робочого листка тіла коду;

– введіть оператор JMP, додаткові модифікатори і мітку;

– виберіть в тілі коду місце, куди доведеться ставити мітку;

– введіть мітку, двокрапки й інструкцію.

Приклад використання стрибка має вигляд:

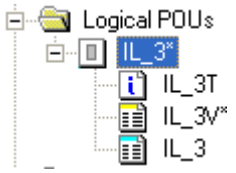
```
LD    var1
EQ    INT#100
JMPC  label
LD    var2
ADD   var3
ST    var4
Label:LD %IX2.2
```

Система програмування може здійснити перекомпіляцію існуючого POU, створеного IL-, FBD- або LD-мовою, кожною з інших двох мов цього переліку.

Для реалізації мовної конверсії:

– переконайтесь, що проект вже побудований і компільований;

– у проектному дереві відкрийте піддерево Logical POUs (Логічні POUs) і позначте ікону існуючого POU;



- клацніть правою клавішею миші по іконі POU, щоб відкрити його контекстне меню;

– виберіть пункт Source conversion (Початок конверсії) і клацніть лівою клавішею. З’явиться діалогове вікно Source conversion (Початок конверсії);

– у діалоговому вікні Source conversion (Початок конверсії), що з’явилося, активізуйте перемикач Overwrite source POU (Переписати початковий POU), якщо бажаєте, визначте нове ім’я POU і виберіть мову перекомпіляції;

– закрийте діалогове вікно і у дереві проекту на місці позначеного POU з’явиться новий POU - конвертований вибраною мовою.

В панелі інструментів клацніть лівою клавішею миші по іконі Compile Worksheet (Компілювати Робочий листок), щоб компілювати POU після конверсії.

Розроблення проекту користувача

Створити IL-мовою програму керування роботою двигуна згідно з алгоритмом наведеним у лабораторній роботі №2.

Для цього, відкрийте діалог New Project (Новий проект), використовуючи пункт меню File (файл). Двічі клацніть по Project Wizard (Майстер проекту) і

пройдіть увесь шлях створення проекту аналогічно варіанту програмування LD-мовою, тобто призначте ім'я проекту та його POU, виберіть у даному випадку IL-мову програмування, бажані імена і типи конфігурації, ресурсу і задачі.

Програмування IL-мовою функціонального блока лічильника для підрахунку кількості натисків на кнопку старту двигуна:

- На робочому полі проектного коду позначте курсором місце початку програмування;
- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group(Група), що з'явиться, виберіть функціональний блок CTU.
- двічі клацніть по ньому і у діалоговому вікні Variable properties (Властивості змінної) змініть типове ім'я CTU-1 на Motor-Count;
- клацніть ОК, діалогове вікно закриється, а на робочому листку проекту залишиться послідовність IL-інструкцій з підказками типів у дужках, яка програмує функціональний блок Motor_Count.

```
LD   (* BOOL *)
ST   Motor_Count.CU
LD   (* BOOL *)
ST   Motor_Count.RESET
LD   (* INT *)
ST   Motor_Count.PV
CAL  Motor_Count
LD   Motor_Count.Q
ST   (* BOOL *)
LD   Motor_Count.CV
ST   (* INT *)
```

Установіть курсор перед коментарем першого рядка і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);

– у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, оголосіть змінну Motor_Start з фізичною адресою вхідного каналу модуля вводу %IX0.0;

- у першому рядку між інструкцією LD і коментарем з'явиться змінна Motor_Start, яка наступною інструкцією ST запам'ятовується у CU-вході лічильника Motor_Count.
- установіть курсор перед коментарем третього рядка, щоб оголосити змінну, яка призначена для RESET(Скид)-входу лічильника Motor_Count;
- в панелі інструментів клацніть по іконі Variable (Змінна) і у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, оголосіть змінну Motor з фізичною адресою вихідного каналу модуля виводу %QX0.0;
- у третьому рядку між інструкцією LD і коментарем з'явиться змінна Motor, яка наступною інструкцією ST запам'ятовується у RESET- вході лічильника Motor_Count;

- установіть курсор перед коментарем п’ятого рядка для визначення параметрів PV - входу;

- в панелі інструментів клацніть по іконі Variable (Змінна), у діалоговому вікні Variable Properties (Властивості змінної), що з’явиться, у рядку Name надрукуйте константу INT#3, натисніть ОК і у п’ятому рядку IL-інструкцій з’являться параметри лічильника;

- установіть курсор перед коментарем дев’ятого рядка, і оголосіть змінну Out, якою керує Q-виход лічильника Motor_Count;

- Після натискання ОК у дев’ятому рядку між інструкцією ST і коментарем з’явиться змінна Out;

- установіть курсор перед коментарем останнього рядка, щоб оголосити змінну, яка запам’ятовує імпульси лічильника Motor_Count;

- в панелі інструментів клацніть по іконі Variable (Змінна), у діалоговому вікні Variable Properties (Властивості змінної), що з’явиться, у рядку Name надрукуйте Pressed, у рядку Usage (Вживання) виберіть VAR, а у рядку Data Type (Тип даних) - INT;

- у рядку Local Variable Groups (Група локальній змінній) активізуйте Default (За замовчуванням) і натисніть ОК, діалогове вікно закриється, а у останньому рядку з’явиться змінна Pressed.

Програмування таймера роботи двигуна:

- Позначте курсором новий рядок для продовження програми;

- активізуйте в панелі інструментів ікону Edit Wizard (Майстер проекту) і у таблиці Group(Група), що з’явиться, виберіть функціональний блок TON;

- двічі клацніть по ньому і у діалоговому вікні Variable properties (Властивості змінної) змініть типове ім’я TON-1 на M_Time;

- натисніть ОК, діалогове вікно закриється, а на робочому листку проекту залишаться IL – інструкції, що програмують таймер M_Time.

```
LD   (* BOOL *)
ST   M_Time.IN
LD   (* TIME *)
ST   M_Time.PT
CAL M_Time
LD   M_Time.Q
ST   (* BOOL *)
LD   M_Time.ET
ST   (* TIME *)
```

За аналогією попереднього функціонального блока оголосіть змінні таймера M_Time, біля операндів LD**,** враховуючи, що на IN-вході таймера змінна Motor, на PT-вході константа часу T#20s, на Q-віході – змінна Mot_time, а ET-виході – змінна Actual_Time.

Програмування непередбаченої зупинки двигуна.

Позначте курсором новий рядок і оголосіть функціональний блок перемикача RS:

На SET- вході оголосіть змінну з ім'ям Out і натисніть ОК. Діалогове вікно закриється, а у першому рядку між інструкцією LD і коментарем з'явиться змінна Out, яка наступною інструкцією ST запам'ятовується у SET - вході перемикача RS-1;

- на RESET1-вході оголосіть змінну Emergency_Stop(Непередбачена зупинка з фізичною адресою вхідного каналу модуля вводу %IX0.1;

- натисніть ОК і у третьому рядку між інструкцією LD і коментарем з'явиться змінна Emergency_Stop (Непередбачена зупинка), яка наступною інструкцією ST запам'ятовується у RESET1- вході перемикача RS-1.

Оскільки двигун має зупинятися не тільки під впливом вхідної змінної Emergency_Stop(Непередбачена зупинка), а ще і після 20 секунд роботи, додамо з клавіатури після змінної Emergency_Stop оператора LD альтернативну вихідну змінну Mot_Time таймера M_Time з оператором OR.

Установіть курсор перед коментарем останнього рядка IL – інструкцій перемикача RS-1 і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);

- у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, виберіть ім'я Motor і натисніть ОК, оскільки змінна Motor вже оголошена;

- діалогове вікно закриється, а у останньому рядку між інструкцією LD і коментарем з'явиться змінна Motor, якою керує Q- вихід перемикача RS-1.

Для компіляції проекту в панелі інструментів натисніть ікону Make (Створювати). Якщо помилки є, відкоригуйте проект, якщо немає - завантажте його, використовуючи відомий шлях: Project Control Dialog → Download→Download.

Перевіряння роботи програми здійсніть в оперативному режимі за допомогою ікони Debug on/off (Налагодження В/В) і симулятора I/O.

При виконанні програми біля змінних різнокольоровим текстом буде висвічуватися їх логічний стан, а при змінній Actual_Time таймера M_Time – секунди затримки.

```
        (*Програмування лічильника кількості натисків
  TRUE    стартової кнопки*)
  TRUE LD  Motor_Start (* BOOL *)
  TRUE ST  Motor_Count.CU
  TRUE LD  Motor       (* BOOL *)
       ST  Motor_Count.RESET
     3 LD  int#3       (* INT *)
       ST  Motor_Count.PV
 FALSE CAL Motor_Count
  TRUE LD  Motor_Count.Q
     0 S   out       (* BOOL *)
     0 LD  Motor_Count.CV
       ST  Pressed     (* INT *)
```

```
  TRUE |  (*Програмування таймера*)
  TRUE |LD  Motor        (* BOOL *)
       |  ST M_Time.IN
20.000 |LD  t#20s        (* TIME *)
       |ST  M_Time.PT
 FALSE |CAL M_Time
 FALSE |LD  M_Time.Q
 4.200 |R   Mot_Time     (* BOOL *)
 4.200 |LD  M_Time.ET
       |ST  Actual_Time (* TIME *)
  TRUE |(*Програмування непердбаченої зупинки двигуна*)
  TRUE |LD  out(* BOOL *)
 FALSE |ST  RS_1.SET
 FALSE |LD  Emergency_Stop(* BOOL *)
 FALSE |OR  Mot_Time
       |ST  RS_1.RESET1
  TRUE |CAL RS_1
  TRUE |LD  RS_1.Q1
```

Завдання для самостійної роботи

1. Розробити програму керування роботою двох двигунів, які після запуску одного з двигунів безперервно по черзі вмикаються і вимикаються. Тривалість роботи двигунів, відповідно, 5 і 10 секунд. Початковий запуск першого двигуна здійснюється одноразовим натисканням пускової кнопки, а початковий запуск другого двигуна – дворазовим. При цьому двигуни можна у будь-яку мить зупинити, а кожне вмикання двигунів підраховується лічильником. Коли кількість вмикань двигунів досягає десяти, їх робота автоматично зупиняється.

2. Розробити програму миготіння світлодіода з одночасним підрахунком кількості вмикань.

Контрольні запитання

1. Як вставити IL-інструкцію у робочий листок тіла коду?
2. Як замінити змінну на таку, що вже оголошена?
3. Як у текстовому редакторі викликати функцію?
4. Як у текстовому редакторі викликати функціональний блок?
5. Як редагуються стрибки в робочих листках тіла коду?
6. До яких мов можна застосувати перекомпіляцію?
7. Як оголосити змінні функціонального блока?
8. Як запрограмувати лічильник кількості натисків стартової кнопки?

Лабораторна робота №5
Програмування ST-мовою у Soflogic-системі KW MULTIPROG

Знайомство з редактором ST – мови

Робочі листки тіла коду редагуються ST–мовою, використовуючи текстовий редактор. Інструкції та вирази користувач може друкувати або

вставляти за допомогою Майстра редагування. Зручніше користуватися Майстром редагування тому, що він містить багато стандартних ключових слів, функцій та функціональних блоків, а це перешкоджає появі помилок при створені ST-програми. При друкуванні інструкцій та виразів рекомендується використовувати абзаци. Кожний рядок починається з номера, а кожна інструкція має закінчуватися крапкою з комою. При використанні Майстра редагування це здійснюється автоматично.

Коментарі створюються у круглих дужках із зірочками. Різні елементи синтаксису мають свій колір: ключові слова - блакитні, змінні та імена екземплярів блоків – чорні, коментарі – зелені.

Для редагування ST-мовою необхідно в проектному дереві відкрити робочий листок тіла коду, подвійно клацаючи по відповідній іконі теки Logical POUs(Логічні POUs).

Якщо Майстра редагування на екрані не видно, натисніть <SHIFT> + <F2> або <Alt>+<3> або в панелі інструментів клацніть по іконі Edit Wizard (Майстра редагування) .

Щоб зробити вставку інструкції необхідно:

– у робочому листку тіла коду визначити курсором місце, де нова інструкція має бути вставлена;

– відкрити вікно Group (Група) Майстра редагування і вибрати Keywords (Ключові слова), де приведено список ключових слів;

– вибрати зі списку бажану інструкцію, наприклад CASE, і двічі клацнути по ній лівою клавішею миші. Синтаксис інструкції автоматично вставиться в робоче поле тіла коду:

```
CASE (*EXPRESSION (must return an INT value*) OF
    (* VALUE  a*):   (*STATEMENTS*);(* VALUE can be a single value  *)
    (* VALUES b*):   (*STATEMENTS*);(* or a set of VALUES          *)
    (*      .    :        .      *) (* for Example:         *)
    (*      .    :        .      *) (*              1   : ....; *)
    (* VALUE  x*):   (*STATEMENTS*);(*              2..4: ....; *)
    ELSE (*STATEMENTS*);
END_CASE;
```

У наведеній мовній структурі замість фактичних змінних і їх значень приведені зеленим текстом коментарі, які оточені круглими дужками і зірочками.

При редагуванні коментарі замінюються фактичними операндами (змінними та значеннями).

Щоб декларувати нову змінну, надрукуйте її ім’я у бажаній позиції тіла коду і встановіть на неї курсор;

– в панелі інструментів клацніть по іконі Variable (Змінна), з’явиться діалогове вікно Variables (Змінні), де у першому рядку поля Name (Ім’я) показане ім’я позначеної змінної;

– виберіть відповідні уставки для змінній (Usage, Data Type та ін.) і у рядку Local Variable Groups (Група локальній змінній) активізуйте Default (За

замовчуванням) і натисніть ОК. Змінна вставиться у робочий листок тіла коду, а її декларація автоматично увійде у робочий листок сітки змінних.

При редагування ST – тіла коду можна використовувати змінні, які вже декларовані і записані у список поля Name (Ім'я). Можна також замінити одну змінну на іншу, яка вже декларована в робочому листку сітки змінних.

Для того, щоб викликати функцію, використовують Edit Wizard (Майстер редагування) . Для цього:

– в робочому листку тіла коду курсором виберіть місце, куди нова функція має бути вставлена;

– відкрийте вікно Group (Група) Майстра редагування і виберіть Functions (Функції);

– клацніть лівою клавішею миші, майстер редагування покаже список функцій;

– виберіть бажану функцію і двічі клацніть по ній, функція автоматично вставиться у позицію, яку показує текстовий курсор;

– замініть зелені коментарі, що оточені круглими дужками і зірочками, необхідними елементами.

Щоб викликати функціональний блок, використовуючи Майстра редагування, необхідно у його вікні Group (Група) вибрати Functions blocks (Функціональні блоки) і подвійно клацнути лівою клавішею миші по бажаному функціональному блоці. У діалоговому вікні Variables (Змінні), що з'явиться, задекларувати його ім'я і натиснути ОК. Мовна структура функціонального блока автоматично вставиться у позначене місце робочого листка тіла коду, а його декларація увійде в групу локальних змінних робочого листка сітки змінних. Після цього, враховуючи підказки, що оточені круглими дужками і зірочками, необхідно надрукувати вхідні та вихідні змінні і константи.

Розроблення проекту користувача

Створити ST-мовою програму керування роботою двигуна згідно з алгоритмом наведеним у лабораторній роботі №2.

Для цього відкрийте діалог New Project (Новий проект), використовуючи пункт меню File (файл). Подвійно клацніть по Project Wizard (Майстер проекту) і пройдіть увесь шлях створення проекту аналогічно варіанту програмування LD-мовою, тобто призначте ім'я проекту та його POU, виберіть у даному випадку ST-мову програмування, бажані імена і типи конфігурації, ресурсу і задачі.

Коли новий проект з'явиться у вікні проектного дерева, можна починати програмування.

Програмування ST-мовою функціонального блока лічильника для підрахунку кількості натисків на кнопку старту двигуна.

На робочому полі проектного коду позначте курсором місце початку програмування і за допомогою Edit Wizard (Майстер проекту) у таблиці Group виберіть функціональний блок CTU;

- задекларуйте цей функціональний блок, як Motor-Count;

- клацніть ОК, діалогове вікно закриється, а на робочому листку проекту залишиться програма функціонального блока лічильника Motor_Count створеного ST- мовою:

```
Motor_Count(CU:=(* BOOL *),RESET:=(* BOOL *),PV:=(* INT *));
(* BOOL *):=Motor_Count.Q;
(* INT *):=Motor_Count.CV;
```

Для оголошення змінних блока Motor_Count:

- установіть курсор перед коментарем CU- входа лічильника і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);

- у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, у рядку Name надрукуйте ім'я змінної Motor_Start, за допомогою якої буде активізуватися двигун;

- заповнити інші рядки, з урахуванням того, що Motor_Start є зовнішньою локальною, дискретною змінною, фізична адреса якої %IX0.0;

- натисніть ОК, діалогове вікно закриється, а CU- входу лічильника Motor_Count присвоюється змінна Motor_Start.

По черзі установіть курсор перед коментарями RESET-входу, PV-входу, Q-виходу, CV- виходу лічильника і оголосіть їх змінні та константу, відповідно, Motor, INT#3, ще раз Motor з фізичною адресою %QX0.0 і Pressed;

Програмування непередбаченої зупинки двигуна:

- позначте курсором новий рядок для продовження програми;

- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, виберіть функціональний блок RS, який має властивості перемикача із домінантою вимикача, і надайте йому ім'я RS_1;

```
RS_1(SET:=(* BOOL *),RESET1:=(* BOOL *));
(* BOOL *):=RS_1.Q1;
```

Щоб оголосити змінні блока RS:

- установіть курсор перед коментарем SET- входу функціонального блока RS і оголосіть внутрішню дискретну змінну Out.

Оскільки двигун має зупинятися після закінчення установленого часу роботи, а також примусово у будь-який момент, для RESET1-входу треба оголосити дві альтернативні змінні. Тоді:

- установіть курсор перед коментарем RESET1-входу і в панелі інструментів клацніть по іконі Variable (Змінна);

- оголосіть першу альтернативну змінну Emergency_Stop (Непередбачена_Зупинка) з фізичною адресою %IX0.1;

- натисніть ОК, діалогове вікно закриється, а RESET1-входу присвоюється перша альтернативна змінна Emergency_Stop;

- надрукуйте поряд з нею логічний оператор OR і в панелі інструментів клацніть по іконі Variable (Змінна);

- у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, оголосіть ім'я другої альтернативної змінної Mot_Time, яка зупинить двигун за командою таймера;

- натисніть ОК і друга альтернативна змінна Mot_Time присвоюється RESET1-входу;

- установіть курсор перед коментарем змінної, яка присвоюється виходу Q1 функціонального блока RS і задекларуйте змінну Motor з адресою вихідного каналу модуля виводу %QX0.0;

Програмування часу роботи двигуна.

Позначте курсором новий рядок для продовження програми;

- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту), у таблиці Group, що з'явиться на екрані, виберіть функціональний блок таймера TON і присвойте йому ім'я M_Time;

```
M_Time(IN:=(* BOOL *),PT:=(* TIME *));
(* BOOL *):=M_Time.Q;
(* TIME *):=M_Time.ET;
```

- оголосіть змінні для IN- входу – Motor, PT- входу – константу T#20s, Q- виходу – Mot_Time і для ET- виходу – Actual_Time.

Програмування лічильника активізацій двигуна:

- Позначте курсором новий рядок для продовження програми;
- активізуйте в панелі інструментів ікону Edit Wizard (Майстер проекту) і у таблиці Group з меню Keywords виберіть оператор IF:

```
IF  (*EXPRESSION (must return a boolean value)*)
    THEN        (*If returned value of EXPRESSION = TRUE*)
    (*STATEMENT*);
END_IF;
```

- установіть курсор після оператора IF і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);

- у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, зі списку Name виберіть вже оголошену змінну Out і натисніть ОК;

- установіть курсор після оператора THEN і надрукуйте вираз Cycle_Count:=Cycle_Count+1;

Для компіляції проекту в панелі інструментів натисніть ікону Make (Створювати).

Якщо помилки є, відкоригуйте проект, якщо немає - завантажте його, використовуючи відомий шлях: Project Control Dialog → Download→Download.

Натисніть кнопку Cold (Холодний) у діалоговому вікні Resource (Ресурс) для холодного запуску симулятора ПЛК;

- в панелі інструментів клацніть по іконі Debug on/off (Налагодження В/В), щоб перейти в оперативний режим роботи. При цьому усі змінні позначаться різними кольорами;

- клацніть лівою клавішею миші по Demoio_Driver, що унизу екрана, для відкриття симулятора I/O;

- тричі подвійно клацніть лівою клавішею миші по нульовому світлодіоду нульового модуля вводу In.

Програма почне виконуватися, а біля змінної Actual_Time таймера M_Time будуть змінюватися секунди затримки. При цьому з кожним новим запуском програми вміст лічильника Cycle_Count збільшується на одиницю.

```
      (*Програмування лічильника кількості натисків стартової
       кнопки*)
FALSE Motor_Count(CU:=Motor_Start,RESET:=Motor,PV:=INT#3);
FALSE out:=Motor_Count.Q;
    0 Pressed(* INT *):=Motor_Count.CV;
      (*Програмування непередбаченої зупинки двигуна*)
FALSE RS_1(SET:=out,RESET1:=Emergency_Stop OR Mot_Time);
 TRUE Motor:=RS_1.Q1;
      (*Програмування таймера часу роботи двигуна*)
 TRUE M_Time(IN:=Motor,PT:=T#20s);
FALSE Mot_Time:=M_Time.Q;
5.700 Actual_Time:=M_Time.ET;
      (*Програмування лічильника активацій двигуна*)
FALSE IF  out
    2     THEN Cycle_Count:=Cycle_Count+1;
      END_IF;
```

Завдання для самостійної роботи

1. Розробити програму керування роботою двох двигунів, які після запуску одного з двигунів безперервно по черзі вмикаються і вимикаються. Тривалість роботи двигунів відповідно 5 і 10 секунд. Початковий запуск першого двигуна здійснюється одноразовим натисненням пускової кнопки, а початковий запуск другого двигуна – дворазовим. При цьому двигуни можна у будь-яку мить зупинити, а кожне вмикання двигунів підраховується лічильником. Коли кількість вмикань двигунів досягає десяти, їх робота автоматично зупиняється.

2. Розробити програму зміни рівня речовини при керуванні ним в автоматичному та ручному режимах. В ручному режимі рівень змінюється під впливом кнопок «Вверх» і «Униз» в діапазоні 0-100%, а в автоматичному – за рахунок позиційного регулювання в діапазоні 40-100%.

Контрольні запитання

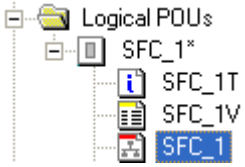
1. Як відкрити робочий листок тіла коду для редагування ST–мовою?
2. Як декларувати нову змінну?
3. Як можна використати вже декларовану змінну при редагуванні тіла коду?
4. Як викликаються функції та функціональні блоки?
5. Як запрограмувати ST-мовою лічильник активізацій двигуна?
6. Як оголосити змінні функціонального блока TON?
7. Навіщо у програмі використовується функціональний блок RS?
8. Як здійснюється налагодження програми?

Лабораторна робота №6
Програмування SFC- мовою у Soflogic-системі KW MULTIPROG

Знайомство з редактором SFC – *мови*

Одною з головних особливостей редагування SFC – мовою є те, що система програмування завжди прагне створити закриту SFC-мережу. Коли вставляються нові кроки і переходи, частина мережі під точкою вставлення автоматично зміщується вертикально вниз. Проте структуру SFC –мережі необхідно організовувати так, щоб був і достатній простір для її розширення убік.

Для редагування робочого листка SFC- коду, необхідно відкрити його подвійним клацанням лівої клавіші миші по відповідній іконі теки Logical POUs(Логічні POUs) у дереві проекту:

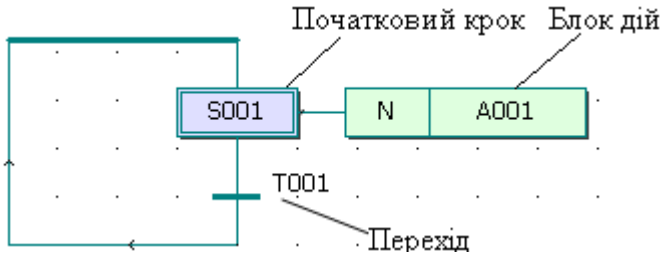


Протягом редагування робочого листка створюється SFC- мережа, яка складається з кроків і переходів, редагованих LD-,FBD-,IL- або ST-мовою. Робочі листки тіла коду кроків і переходів розташовуються у відповідній піддиректорії проектного дерева і називаються деталлю.

Для створення SFC- мережі у робочому листку тіла коду курсором позначте місце вставки;

– в панелі інструментів клацніть по іконі Create step transition sequence (Вставити послідовність крок-перехід) і на екрані з'явиться SFC- мережа з одним кроком і одним переходом. Перший вставлений крок автоматично є початковим кроком, він має подвійну рамку.

–



Щоб розширити існуючу мережу за рахунок нових кроків і переходів:

– клацніть лівою клавішею миші по бажаному кроці, щоб позначити місце розширення мережі;

– в панелі інструментів клацніть по іконі Create step transition sequence (Вставити послідовність крок - перехід) і інша пара (крок-перехід) з'явиться між позначеним кроком і наступним переходом.

Якщо позначити перехід замість кроку і вставити нову послідовність (крок – перехід), то вона вставиться нижче позначеного переходу.

Ширина кроків і дій SFC-мережі завжди однакова, якщо у підменю Loyout (Розташування) альтернатива Fixed SFC objects Width (Постійна ширина SFC об'єкта)- активізована. Якщо ні, ширина їх коригується автоматично відповідно до довжини імені кроку або дії.

При необхідності можна змінити початковий крок на звичайний, а звичайний – на початковий. Для цього:

– двічі клацнути лівою клавішею миші по кроці, який бажано змінити. З'явиться діалог Step (Крок);

– активізувати перемикач Initial step (Початковий крок), щоб змінити звичайний крок на початковий;

– вимкнути перемикач Initial step (Початковий крок), щоб змінити початковий крок на звичайний. Звичайний крок зображується в робочому листку прямокутником з одинарною рамкою.

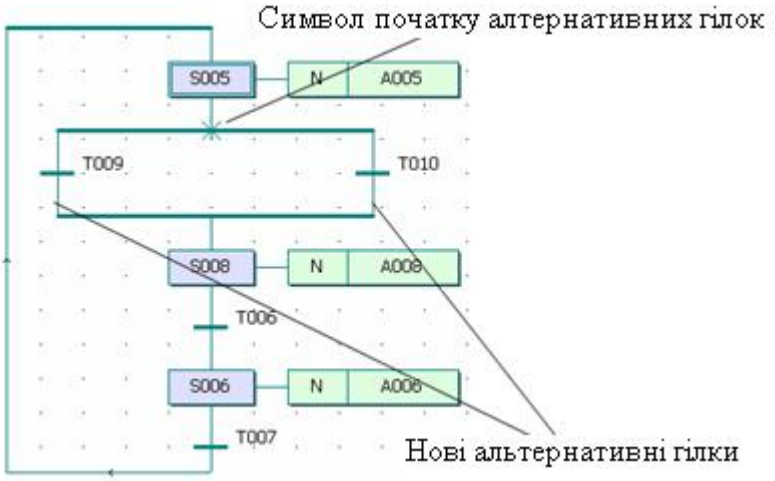
Протягом редагування можна вставити у SFC–мережу альтернативні або паралельні гілки.

Щоб вставити альтернативні гілки:

– позначте крок, де бажано вставити гілки;

– в панелі інструментів клацніть по іконі Insert Simultaneous/Alternativ Branches (Вставка Одночасної/Альтернативної гілки). З'явиться діалог Divergence (Розходження);

– введіть бажану кількість гілок у вікні діалогу Branches Count (Рахунок гілок) і натисніть ОК.

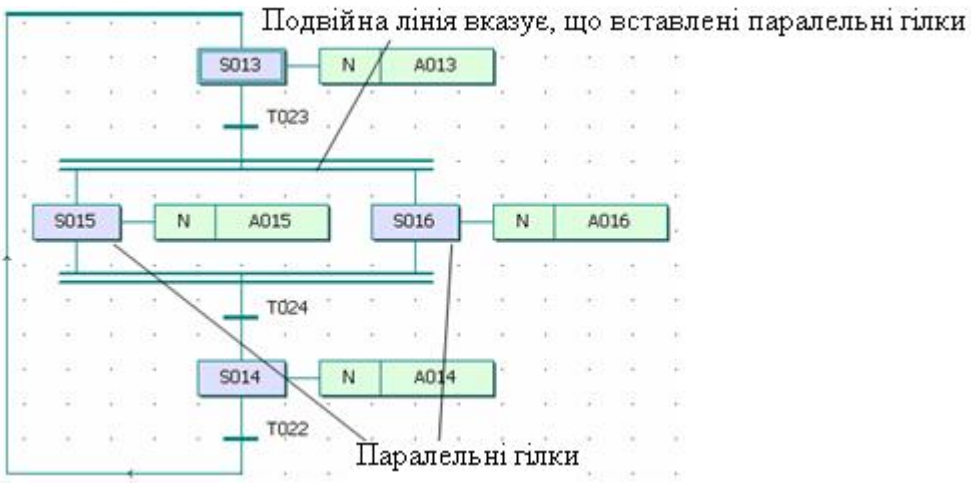


Щоб вставити паралельні гілки:

– позначте перехід, де бажано вставити гілки;

– клацніть по іконі Insert Simultaneous/Alternativ Branches (Вставка Одночасної/Альтернативної гілки);

– введіть необхіднукількість гілок у вікні діалогу Branches Count (Рахунок гілок) і натисніть ОК:



Вставлення гілок можна здійснювати, використовуючи режим редагування SFC – гілок.

Щоб вставити паралельні гілки:

- в панелі інструментів клацніть лівою клавішею миші по іконі Insert SFC branch (Вставка SFC – гілки). З'явиться курсор у вигляді SFC – гілки;

– спрямуйте курсор на перехід, після якого має бути розходження гілок і клацніть лівою клавішею миші. З'явиться подвійна лінія. Перемістіть курсор вбік на вільне місце, де бажаєте встановити новий крок з блоком дії;

– клацніть лівою клавішею миші, щоб зафіксувати верхню подвійну лінію;

– перемістіть курсор на перехід, де бажано встановити нижню подвійну лінію, і клацніть лівою клавішею миші.

Режим редагування SFC – гілок завершується автоматично після того, як нова гілка вставлена. Для вставлення іншої гілки повторно виберіть режим редагування SFC - гілок, знову клацаючи по іконі Insert SFC branch (Вставка SFC – гілки).

Щоб вставити альтернативну гілку, треба повторити такі самі дії, як і у попередньому варіанті, тільки спочатку спрямуйте курсор у вигляді SFC – гілки не на перехід, а на крок.

В існуючих SFC – мережах можна вставити стрибки. Стрибки вказують шлях виконання програми у розімкненій SFC-мережі. Стрибок має ім'я мети, куди він прямує. Щоб вставити стрибок:

– двічі клацніть лівою клавішею миші по кроці, з якого має бути стрибок. З'явиться діалог Step (Крок);

– у рядку Name(Ім'я) введіть ім'я кроку, на який здійснюється стрибок;

– активізуйте перемикач Jump (Стрибок) і натисніть ОК;

– з'явиться діалог Jump/End step insert control (Стрибок/Кінець управління кроком вставки);

– активізуйте перемикач Show marked objects for delete (Показ позначених об'єктів для видалення) і натисніть ОК, щоб побачити усі об'єкти SFC-мережі, що видаляються при вставленні стрибка;

– знову викличте діалог Jump/End step insert control (Стрибок/Кінець управління кроком вставки);

– натисніть кнопку Change (Заміна) для видалення усіх непотрібних об'єктів і вставлення стрибка з іменем мети.

Можна вставити кінцеві кроки, якщо бажаєте закінчити SFC - мережу.

Для цього:

– клацніть по кроці, який бажаєте зробити кінцевим. З'явиться діалог Step (Крок);

– активізуйте перемикач End step (Кінцевий крок) і натисніть ОК;

– з'явиться діалог Jump/End step insert control (Стрибок/Кінець управління кроком вставки);

- активізуйте перемикач Show marked objects for delete (Показ позначених об'єктів для видалення) і натисніть ОК, щоб побачити усі об'єкти SFC-мережі, що видаляються при вставленні стрибка;

– знову викличте діалог Jump/End step insert control (Стрибок/Кінець управління кроком вставки);

– натисніть кнопку Change (Заміна) для видалення усіх непотрібних об'єктів і вставлення кінцевого кроку.

Можна вставити блок дій до кроку, якщо він був видалений. Можна додати нові блоки дій до одного існуючого і створити, таким чином, у робочому листку тіла коду крок із складеним блоком дій. Щоб видалити існуючий блок дій, клацніть по ньому лівою клавішею миші і натисніть «Delete» на клавіатурі комп'ютера.

Щоб вставити новий блок дії:

- позначте крок, до якого бажаєте вставити новий блок дії або позначте існуючий блок дії, якщо бажаєте мати складений блок дій;

– в панелі інструментів клацніть по іконі Create action (Створити дію). З'явиться діалогове вікно Action Properties (Властивості дії);

– введіть нове ім'я дії, якщо бажаєте, і натисніть ОК. Новий блок дії з'явиться у робочому листку тіла коду.

Робочий листок тіла коду дії, так звану деталь, можна присвоїти будь-якому кроку існуючої SFC – мережі.

Для цього:

– позначте крок, якому бажаєте присвоїти нову деталь;

– в панелі інструментів клацніть по іконі Create action (Створити дію);

– з'явиться діалог Action Properties (Властивості дій);

– у першому рядку поля Name (Ім'я) введіть ім'я нового робочого листка тіла коду (деталі);

– активізуйте кнопку Detail (Деталь) і натисніть ОК. Позначений крок отримує нову дію з власним іменем.

Дії можуть мати логічну змінну або тіло коду.

Протягом редагування проекту можна змінити властивості кроку* і *блоку дії, а також переходу.

Для зміни властивостей кроку:

– двічі клацніть по кроку лівою клавішею миші. З'явиться діалогове вікно Step(Крок);

– змініть ім'я або тип кроку і натисніть ОК.

Щоб змінити властивості блока дій:

– правою клавішею миші клацніть по блоку дії і виберіть пункт контекстного меню Objects Properties (Властивості об'єкту). З'явиться діалогове вікно Action Properties (Властивості дії);

– зробіть зміну імені або класифікатора і натисніть ОК.

Для зміни властивостей переходу:

– клацніть правою клавішею миші по переходу, щоб викликати контекстне меню, і виберіть пункт Objects Properties (Властивості об'єкта). З'явиться діалог Transition (Перехід);

– змініть ім'я або тип переходу і натисніть ОК.

Переходи можуть мати деталі або прямі зв'язки. Коли перехід має власне тіло коду, тобто деталь, у діалозі Transition (Перехід) треба активізувати тип Detail (Деталь).

Прямий зв'язок означає, що у перехода немає тіла коду, і він безпосередньо з'єднується з LD – або FBD – мережею, або має змінну. У цьому випадку в діалозі Transition (Перехід) потрібно активізувати тип Direct connection (Прямий зв'язок). У разі прямого зв'язку перехід не має імені.

Для створення деталі дії або переходу в робочому листку SFC-коду подвійно клацніть по дії або переходу, з'явиться діалогове вікно Insert (Вставити);

– виберіть мову програмування деталі;

– натисніть ОК і робочий листок дії або переходу відкриється для редагування і одночасно з'явиться у проектному дереві.

Дії в робочих листках тіла коду можуть бути зображені логічними змінними.

Щоб присвоїти дії ім'я змінної або замінити на ім'я змінної, яка вже декларована:

– правою клавішею миші клацніть по дії, щоб відкрити контекстне меню;

– виберіть пункт Object Properties (Властивості об'єкту). З'явиться діалог Action Properties (Властивості дії);

– у діалоговій сторінці Action (Дія) активізуйте кнопку Variable (Змінна);

– зі списку поля Name (Ім'я), що містить усі оголошені змінні, виберіть бажану і клацніть по ній лівою клавішею миші;

– у рядку Local Variable Groups (Група локальній змінній) активізуйте Default (За замовчуванням);

– уатисніть ОК і змінна буде вставлена у блок дії.

Щоб декларувати нову змінну дії:

– клацніть по дії правою клавішею миші, щоб відкрити контекстне меню;

– виберіть Object Properties (Властивості об’єкту) і клацніть лівою клавішею миші. З’явиться діалог Action Properties (Властивості дії);

– у діалоговій сторінці Action (Дія) активізуйте перемикач Variable (Змінна);

- у першому рядку поля Name (Ім’я) надрукуйте ім’я дії, виберіть відповідні установки для змінної.

Після вибору компетенції клацніть ОК, змінна вставиться у робочий листок тіла коду. Автоматично її декларація потрапляє у робочий листок відповідної сітки змінних. У разі глобальної змінної декларація вставляється у сітку локальних змінних, використовуючи ключове слово VAR_EXTERNAL, а у робочий листок сітки глобальних змінних, використовуючи VAR_GLOBAL.

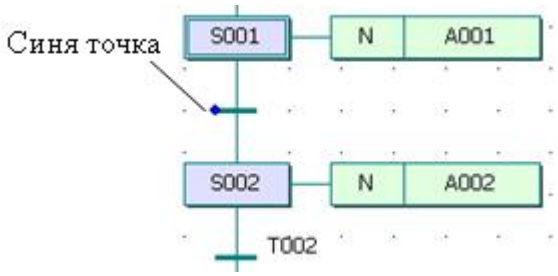
У разі декларування змінних переходів можна вставити змінну вже приєднаною до переходу або з’єднати з переходом змінну, яка встановлена у будь-якому місці робочого листка тіла коду.

Щоб присвоїти переходу ім’я вже декларованої змінної або надати йому нове ім’я, або замінити існуюче ім’я переходу:

– клацніть по переходу правою клавішею миші, щоб відкрити контекстне меню;

– виберіть пункт Object Properties (Властивості об’єкта). З’явиться діалогове вікно Transition (Перехід);

– активізуйте перемикач Direct connection (Прямий зв'язок) і натисніть ОК. Перехід показується синьою точкою з’єднання.



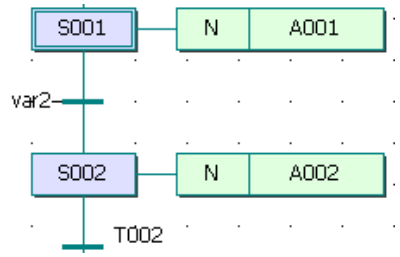
Позначте перехід, лівою клавішею миші;

– в панелі інструментів клацніть по іконі Variable (Зміна), з’явиться діалогове вікно Variable Properties (Властивості змінної);

– у вікні вже декларованих змінних поля Name (Ім’я) клацніть по бажаному імені, і воно увійде у перший рядок. Якщо немає - надрукуйте нове ім’я.

Коли переходу надаєте нове ім’я у діалоговому вікні Variable Properties (Властивості змінної) виберіть відповідні установки і визначтесь з компетенцією змінної.

Закрийте діалогове вікно і змінна потрапить у робочий листок тіла коду, а її декларація автоматично потрапить у відповідний робочий листок сітки змінних:



Замість змінних до переходу можуть бути приєднані функції або LD- мережі за допомогою прямих зв'язків. У цих випадках спочатку необхідно в робоче поле тіла коду вставити функцію чи LD- мережу, а потім з'єднати її з точкою зв'язку переходу, активізуючи в панелі інструментів ікону Connect Objects (З'єднати об'єкти).

При видаленні фрагментів SFC-коду, в які входять більш ніж один крок і один перехід, або цілі гілки, на робочому полі тіла коду іноді залишаються дуже довгі лінії сполучення. У цих випадках довжину ліній можна оптимізувати автоматично. Для цього:

– позначте за допомогою лівої клавіші миші частину лінії, яку потрібно оптимізувати;

– у підменю Edit>Stretch/Compress (Редагування>Розтягання/ Стиснення) виберіть пункт Optimize SFC Lines (Оптимізація SFC –лінії) і довжина ліній оптимізується автоматично.

Розроблення проекту користувача

Створити SFC-мовою програму керування роботою двигуна згідно з алгоритмом наведеним у лабораторній роботі №2.

Для цього відкрийте діалог New Project (Новий проект), використовуючи пункт меню File (Файл). Двічі клацніть по Project Wizard (Майстер проекту) і пройдіть увесь шлях створення проекту аналогічно варіанту програмування LD-мовою, тобто призначте ім'я проекту та його POU, виберіть у даному випадку SFC-мову програмування, бажані імена і типи конфігурації, ресурсу і задачі.

Будь-який проектний код, що створено SFC – мовою, складається з кроків і переходів, об'єднаних SFC – схемою. Тому перед тим, як їх програмувати необхідно побудувати структуру цієї схеми.

Створення SFC – схеми керування роботою двигуна:

Спочатку побудуємо послідовну мережу кроків та переходів, які забезпечують ініціалізацію змінних, програмування умов вмикання в роботу двигуна, програмування перемикача для створення можливості непередбаченого вимкнення двигуна та лічильника кількості активізацій двигуна.

На робочому полі проектного коду позначте курсором місце початку програмування;

- в панелі інструментів клацніть лівою клавішею миші по іконі Creat step transition sequence(Створити послідовні крок і перехід). На робочому полі

з’явиться початкова SFC-схема з одного кроку S001 і дії A001, одного переходу T001, а також повернення назад.

Якщо з’явився не початковий крок, а звичайний, тобто прямокутник кроку створений не подвійною лінією, а одинарною, то перетворить його у початковий:

Для призначення імені компонентам SFC-схеми :

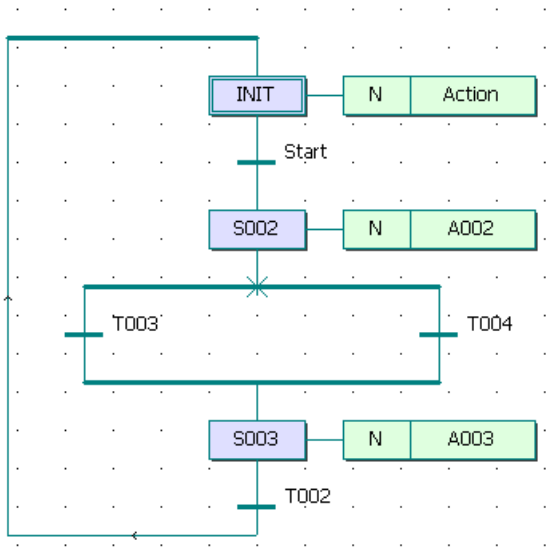
- клацніть правою клавішею миші по кроку S001 і в меню, що з’явиться, лівою клавішею миші активізуйте Object Properties… У вікні Step у рядку Name надрукуйте ім’я INIT і клацніть ОК;

- таке саме зробіть послідовно з дією A001 та переходом T001, привласнюючи їм імена відповідно Action з класифікатором (Qualifier) N замість A001 у діалоговому вікні Action Properties (Властивості дії) та Start замість T001 у діалоговому вікні Transition (Перехід), при цьому в обох вікнах альтернатива Detail (Деталь) має бути активізована.

Для створення другої пари (крок і перехід) клацніть лівою клавішею миші спочатку по переходу Start, а потім в панелі інструментів по іконі Creat step transition sequence(Створити послідовні крок і перехід). В наслідок цього існуюча SFC-схема збільшиться ще на один крок S002 і перехід T002. При цьому присвоювати імена їм поки не будемо, оскільки далі на базі другої пари кроку і переходу створимо у SFC-схемі розгалуження.

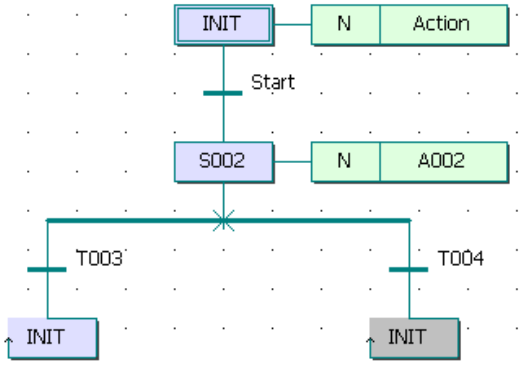
Для створення альтернативних гілок розгалуження позначте курсором другий крок S002 SFC-схеми і в панелі інструментів клацніть по іконі Insert Simultaneous/Alternative Divergence (Вставити Паралельне/Альтернативне розгалуження);

- у діалоговому вікні Divergence (Розгалуження), що з’явиться, у полі Branches Count (Кількість гілок) надрукуйте 2 і клацніть ОК. Проектний код у робочому полі набуває вигляд:



У лівій та правій гілках альтернативного розгалуження створимо стрибки на початковий крок INIT. Для цього:

- позначте курсором перехід T003 і в панелі інструментів клацніть по іконі Creat step transition sequence (Створити послідовні крок і перехід), у лівій гільці з'являться крок S004 з дією A004 і перехід T005;
- правою клавішею миші клацніть по кроку S004 і в меню, що з'явиться, лівою клавішею миші виберіть Object Properties… У полі Type активізуйте альтернативу Jump(Стрибок) і клацніть ОК;
- у діалоговому вікні Jump/End step insert control (Стрибок/Кінець управління кроком вставки), що з'явиться, натисніть Change(Змінити) і в лівій гілці залишаться тільки перехід T003 і безадресний стрибок S004 у вигляді прямокутника зі стрілкою;
- щоб призначити адресу стрибка S004 лівою клавішею миші подвійно клацніть по ньому і у діалоговому вікні Step(Крок), що з'явиться, вкажіть у полі Name ім'я адреси – початковий крок INIT. Натисніть ОК і у прямокутнику стрибка замість S004 з'явиться INIT;
- такі самі перетворення, тільки на базі переходу T004, зробіть і з правою гілкою SFC-схеми. Після цього SFC-схема набуває вигляд:



- призначте новим компонентам SFC-схеми, що з'явилися на робочому полі (крок S002, дія A002 і переходи T003 і T004), імена відповідно Step1, Action1, Trans і Trans1, користуючись вже описаними раніше правилами.

Програмування кроків і переходів:

Після створення SFC-схеми необхідно запрограмувати дії кроків і умови переходів відповідно до алгоритму керування роботою двигуна.

Згідно з принципами SFC – мови, перший крок є ініціалізаційним. Тому змінній керування роботою двигуна Motor необхідно присвоїти значення FALSE, а лічильник скинути. Для цього:

- лівою клавішею миші двічі клацніть по дії початкового кроку Action, у діалоговому вікні Insert (Вставлення), що з'явиться, виберіть ST-мову програмування і натисніть ОК. Відкриється робоче поле дії Action;
- користуючись синтаксисом ST-мови присвойте змінній Motor, що керує роботою двигуна, значення FALSE, тобто надрукуйте: Motor :=FALSE;
- перейдіть на новий рядок і в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту), у таблиці Group, що з'явиться на екрані, виберіть меню Keywords;
- подвійно клацніть по оператору IF і на робочому листку проекту з'явиться його синтаксис з відповідним коментарем;

- установіть курсор після оператора IF і надрукуйте вираз Cycle_Count>10;

- установіть курсор після оператора THEN і надрукуйте вираз Cycle_Count:=Cycle_Count+1, який рахує кількість активізацій двигуна.

Створений фрагмент програми дії Action забезпечує обнуління лічильника після 10 активізацій двигуна.

Після цього ST-програма дії Action набуває завершений вигляд:

```
motor:=FALSE;
IF Cycle_Count>10   (*EXPRESSION (must return a boolean value)*)
    THEN Cycle_Count:=0(*If returned value of EXPRESSION = TRUE*
    (*STATEMENT*);
END_IF;
```

Щоб оголосити ці змінні, установіть курсор спочатку на ім’я Motor, а потім на Cycle_Count і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);

- у діалогових вікнах Variables (Змінні), що з’являться, заповнити необхідні рядки, для кожної локальної дискретної змінної, з урахуванням фізичної адреси змінної Motor – %QX0.0;

Програмування переходу Start:

Умовою початку роботи двигуна є триразове натискання стартової кнопки. Запрограмуємо її LD-мовою, використовуючи прямий зв'язок, коли перехід не має тіла коду, а безпосередньо сполучається з LD-мережею. У даному випадку перехід не повинен мати ім’я.

Щоб реалізувати цей намір:

- клацніть правою клавішею миші по переходу Start, у діалоговому вікні Trasition (Перехід), що з’явиться, активізуйте альтернативу Direct Connection (Безпосереднє сполучення), рядок Name (Ім’я) зробіть порожнім і натисніть ОК;

- діалогове вікно закриється, а у робочому полі SFC - програми після початкового кроку INIT залишиться безіменна смужка переходу з блакитною точкою;

- для створення умови переходу LD-мовою клацніть лівою клавішею миші у зручному місці робочого поля зліва від безіменного переходу;

- в панелі інструментів клацніть по іконі Contact network (Мережа контактів) і в установленому місці робочого поля з’явиться LD- мережа 001 певної ширини з контактом і котушкою;

- вилучите з LD- мережі праву шину живлення і котушку C001 послідовно помічаючи їх лівою клавішею миші і натискаючи клавішу Delet на клавіатурі ЕОМ;

- в панелі інструментів активізуйте іконку Connect objects (З’єднати об’єкти) і за допомогою курсору з’єднайте прямий контакт C000 LD- мережі з блакитною точкою смужки безіменного переходу;

- лівою клавішею миші позначте лінію, що сполучає прямий контакт і перехід;

- в панелі інструментів клацніть по іконі Edit Wizard (Майстер редагування) і з таблиці Group (Група), що з'явиться на екрані, виберіть функціональний блок лічильника CTU і присвойте йому ім'я Motor-Count;

- клацніть на функціональному блоці CTU по синій точці Reset-входу;

- в панелі інструментів клацніть по іконі Add contact left (Додати контакт зліва) і на Reset -вході лічильника з'явиться контакт C002;

- в панелі інструментів активізуйте кнопку Connect objects (З'єднати об'єкти);

- клацніть по контакту C002 і перемістіть його курсором ліворуч до шини живлення 001 так, щоб він опинився під контактом C000;

- в панелі інструментів клацніть по іконі Mark (Позначка), щоб деактивізувати кнопку Connect objects (З'єднати об'єкт).

Оголошення властивостей прямих контактів LD- мережі:

- двічі клацніть лівою клавішею миші по контакту C000 і оголосіть змінну Motor_Start з фізичною адресою %IX0.0;

- двічі клацніть по контакту C002 і у діалоговому вікні Contact/Coil Properties (Властивості Контакт/котушка), що з'явиться, з існуючого списку вже оголошених змінних виберіть Motor і клацніть ОК. На робочому листку з'явиться змінна Motor замість C002;

- для PV-входу введіть константу INT#3, щоб двигун (Motor) спрацював після триразового натиснення стартової кнопки;

- для CV-виходу лічильника, задекларуйте змінну Pressed, яка запам'ятовує поточне значення лічильника.

Програмування дії Actin1 кроку Step1:

Після триразового натискання кнопки Motor_Start має працювати двигун. Тому у наступному кроці Step1 запрограмуємо роботу двигуна, передбачивши можливість його екстреного зупинення. Для цього:

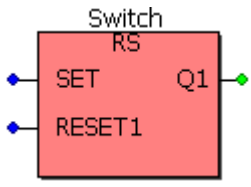
- лівою клавішею миші двічі клацніть по дії першого кроку Action1, у діалоговому вікні Insert (Вставлення), що з'явиться, виберіть FBD-мову програмування і натисніть ОК;

- у робочому полі дії Action1, що з'явиться, лівою клавішею миші позначте місце створення програмного коду;

- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, виберіть функціональний блок перемикача з домінантою вимкнення RS;

- двічі клацніть по ньому і у діалоговому вікні Variable properties (Властивості змінної) змініть типове ім'я RS -1 на Switch;

- клацніть ОК, діалогове вікно закриється, а у робочому листку дії Action1 залишиться функціональний блок перемикача Switch створений FBD-мовою:



Оголошення змінних блока Switch треба здійснити з урахуванням того, що:

- біля SET- входу має бути ім'я виходу лічильника кількості натисків стартової кнопки - Motor_Count.Q;
- біля RESET1-входу – ім'я зовнішньої змінної Emergency_Stop (Непередбачене зупинення) з фізичною адресою модуля вводу %IX0.1;
- біля Q1- виходу – ім'я вже оголошеної змінної Motor.

Закрийте робоче поле дії Action1 кроку Step1, а його вміст запам'ятайте, натиснувши відповідну кнопку у діалоговому вікні MULTIPROG.

Разом з початком роботи двигуна має працювати лічильник його активізацій. Тому для першого кроку створимо ще одну дію, яка має виконуватися разом з дією Action1. Для цього:

- на SFC-схемі лівою клавішею миші клацніть по прямокутнику дії Action1, а в панелі інструментів по іконі Create action (Створення дії). Поряд з прямокутником дії Action1 з'явиться прямокутник іншої дії, якому присвойте ім'я Counter, керуючись вже відомою схемою.

Програмування дії Counter кроку Step1:

Створимо лічильник, який рахує кожне вмикання двигуна. При цьому після 10 циклів роботи показання лічильника скидаються у нуль.

- Лівою клавішею миші двічі клацніть по дії Counter;
- у діалоговому вікні, що з'явиться, виберіть ST-мову, закрийте вікно і на екрані з'явиться робоче поле для програмування лічильника;
- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, виберіть альтернативу Keywords (Ключові слова);
- двічі клацніть по оператору IF і у робочому полі з'явиться його синтаксис з відповідним коментарем:

```
IF  (*EXPRESSION (must return a boolean value)*)
    THEN        (*If returned value of EXPRESSION = TRUE*)
    (*STATEMENT*);
END_IF;
```

Оголошення змінних дії Counter:

- Установіть курсор після оператора IF і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);
- у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, у рядку Name надрукуйте ім'я виходу лічильника кількості натисків стартової кнопки - Motor_Count.Q і натисніть ОК;
- діалогове вікно закриється, а біля оператора IF з'явиться логічна змінна Motor_Count.Q;
- установіть курсор після оператора THEN і надрукуйте вираз Cycle_Count:=Cycle_Count+1.

Оскільки змінна Cycle_Count вже оголошена, лічильник активізацій двигуна є створеним. Закрийте робоче поле переходу, а його код запам'ятайте.

Програмування переходу Trans:

Щоб показання лічильника скидалися після 10 активізацій двигуна створимо IL – мовою відповідну програму переходу Trans . Для цього:

- лівою клавішею миші двічі клацніть по переходу Trans;
- у діалоговому вікні, що з'явиться, виберіть IL-мову, закрийте його і на екрані з'явиться робоче поле для програмування;
- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані, в альтернативі Operators (Оператори) виберіть оператор GT (Більш ніж);
- двічі клацніть по ньому і на робочому полі переходу Trans з'явиться синтаксис цього оператора:

```
LD  var_1
GT  var_2
ST  var_out
```

Замість var1 надрукуйте вже оголошену змінну Cycle_Count, замість var2 – константу 10, а замість var_out – ім'я переходу Trans;

- установіть курсор на ім'я переходу Trans і клацніть лівою клавішею миші, з'явиться діалогове вікно Variables, у рядку Name якого надруковано ім'я переходу Trans;
- лівою клавішею миші клацніть по OK;
- закрийте робоче поле переходу Trans і запам'ятайте створену програму.

Програмування часу роботи двигуна:

Оскільки двигун після вмикання працює 20 секунд, умовою переходу Trans1 має бути саме 20с, які відлічує таймер.

Для програмування таймера ST- мовою:

- лівою клавішею миші двічі клацніть по переходу Trans1;
- у діалоговому вікні, що з'явиться, виберіть ST-мову, закрийте вікно і на екрані з'явиться робоче поле для програмування таймера;
- в панелі інструментів активізуйте ікону Edit Wizard (Майстер проекту) і у таблиці Group, що з'явиться на екрані виберіть функціональний блок таймера TON і оголосіть його екземпляр, як Motor_Time;

```
Motor_Time(IN:=(* BOOL *),PT:=(* TIME *));
(* BOOL *):=Motor_Time.Q;
(* TIME *):=Motor_Time.ET;
```

- установіть курсор перед коментарем IN- входу функціонального блока Motor_Time і в панелі інструментів лівою клавішею миші клацніть по іконі Variable (Змінна);
- у діалоговому вікні Variable Properties (Властивості змінної), що з'явиться, у рядку Name надрукуйте ім'я Step1.X, яке відповідає назві кроку Step1 і натисніть OK;
- діалогове вікно закриється, а IN- входу функціонального блока таймера Motor_Time присвоюється змінна Step1.X;
- установіть курсор перед коментарем PT- входу і задекларуйте константу часу T#20s, яка встановлює час роботи двигуна;

- установіть курсор перед коментарем змінної, яка присвоюється Q-виходу таймера і задекларуйте ім’я переходу Trans1;

- установіть курсор перед коментарем ET- виходу таймера і оголосіть ім’я внутрішньої змінної Act_Time, яка має відображати фактичний час роботи двигуна;

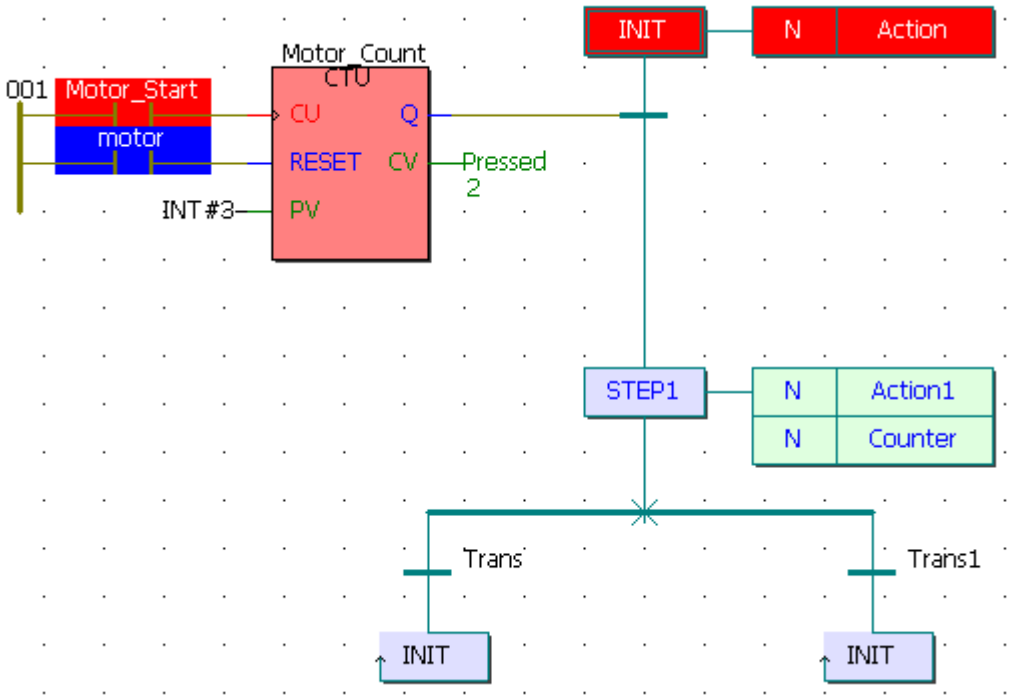
Таким чином, відповідно до створеної програми після триразового натискання кнопки Motor_Start вмикається двигун і лічильник його активізацій. Оскільки у лівій гілці альтернативного розгалуження умови для переходу Trans відбуваються тільки після 10 вмикань двигуна програма виконується відповідно до умов переходу Trans1 правої гилки розгалуження, тобто після 20 секунд роботи двигуна у виконанні програми здійснюється стрибок на початковий крок і робота двигуна зупиняється. Для повторного запуску двигуна необхідно знову тричі натиснути стартову кнопку. Після десятої активізації роботи двигуна виконуються умови переходу Trans лівої гілки розгалуження і стрибок на початковий крок вже здійснюється через неї. Це призводить не тільки до зупинки двигуна, а і до обнуління лічильника його активізацій.

Після створення проектного коду в панелі інструментів натисніть ікону Make (Створювати), для компіляції проекту.

Якщо помилки є, відкоригуйте проект, якщо немає - завантажте його, використовуючи відомий шлях: Project Control Dialog → Download→Download.

Перевірку роботи програми здійснити за допомогою стимулятора, як це відбувалось у попередніх лабораторних роботах.

Програма почне виконуватися, про що свідчить перехід червоного кольору від початкового кроку INIT до кроку STEP1 після триразового натискання на нульовий світлодіод нульового модуля вводу симулятора.



Після затримки на 20 секунд червоний колір від кроку STEP1 через перехід Trans1 перестрибує на початковий крок INIT. При цьому з кожним новим запуском програми у дії Counter змінна лічильника Cycle_Count збільшується на одиницю. Для екстреної зупинки двигуна необхідно натиснути на перший світлодіод нульового модуля вводу. Коли кількість активізацій двигуна перевищує цифру 10, виконання програми продовжується через перехід Trans лівої гилки розгалуження на початковий крок INIT. При цьому робота двигуна зупиняється, а лічильник скидається у нуль.

Завдання для самостійної роботи

1. Розробити програму керування роботою двох двигунів, які після запуску одного з двигунів безперервно по черзі вмикаються і вимикаються. Тривалість роботи двигунів відповідно 5 і 10 секунд. Початковий запуск першого двигуна здійснюється одноразовим натисненням пускової кнопки, а початковий запуск другого двигуна – дворазовим. При цьому двигуни можна у будь-яку мить зупинити, а кожне вмикання двигунів підраховується лічильником. Коли кількість вмикань двигунів досягає десяти, їх робота автоматично зупиняється.
2. Розробити програму керування роботою світлофора, що працює на перехресті доріг. Перехід із зеленого кольору на жовтий у світлофорі відбувається після триразового миготіння зеленого.

Контрольні запитання

1. Як створити першу SFC-мережу?
2. Як розширити існуючу мережу?
3. Як змінити початковий крок на звичайний, а звичайний – на початковий?
4. Як вставити в SFC-мережу альтернативні гілки?
5. Як вставити в SFC-мережу паралельні гілки?
6. Як користуватися режимом редагування SFC – гілок?
7. Як вставити в SFC-мережу стрибки?
8. Як додати нові блоки дій до одного існуючого?
10. Як змінити властивості кроку, блока дії, переходу?
11. Як створити деталь дії або переходу?
12. Як до переходу можуть бути приєднані функції або LD- мережі?
14. Як видалити фрагменти SFC-коду?
15. Як здійснюється програмування кроків і переходів SFC-схеми?
16. Коли у режимі налагодження елементи SFC-схеми позначаються червоним кольором?

Основні відомості з синтаксису графічних мов програмування

LD –мова.

1. ***Прямий контакт:***

Ліве сполучення Праве сполучення

Стан правого кінця лінії сполучення є логічним ’I’ (AND) між станом лівого кінця і значенням змінної, асоційованої з контактом.

2. ***Інвертований контакт:***

Ліве сполучення Праве сполучення

Стан правого кінця лінії сполучення є логічним ’I’ (AND) між станом лівого кінця і запереченням значення змінної, асоційованої з контактом.

3. ***Пряма котушка:***

Ліве сполучення Праве сполучення

Змінній котушки призначається стан лівого сполучення. Стан лівого сполучення розповсюджується на праве сполучення.

4. ***Інвертована котушка:***

Ліве сполучення Праве сполучення

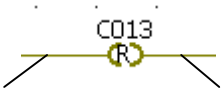
Змінній котушки призначається заперечення стану лівого сполучення. Стан лівого сполучення розповсюджується на праве сполучення. Праве сполучення може бути підключене до правої вертикальної шини живлення.

5. ***'Set' котушка:***

Ліве сполучення Праве сполучення

Асоційована змінна встановлюється в TRUE (SET TO TRUE), коли стан лівого сполучення стає рівним TRUE. Вихідна змінна утримує цей стан до тих пір, поки вона не буде інвертована котушкою 'RESET'. Стан лівого сполучення розповсюджується на праве сполучення.

6. ***"Reset" котушка:***

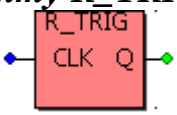


Ліве сполучення Праве сполучення

Асоційована змінна встановлюється в FALSE (RESET TO FALSE), коли стан лівого сполучення стає рівним TRUE. Вихідна змінна утримує цей стан до тих пір, поки вона не буде інвертована котушкою "SET". Стан лівого сполучення розповсюджується на праве сполучення.

FBD –мова.

7. ***Детектор переднього фронту R_TRIG***



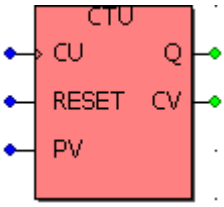
CLK	BOOL	будь-яка булева змінна.
Q	BOOL	TRUE, коли CLK змінюється з FALS на TRUE. У решті випадків – FALS.

8. ***Детектор заднього фронту F_TRIG***

CLK	BOOL	будь-яка булева змінна.
Q	BOOL	TRUE, коли CLK змінюється з TRUE на FALS. У решті випадків – FALS.

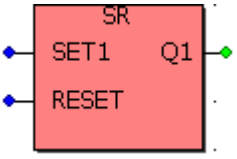
Попередження: Блок F_TRIG має властивість формувати хибний імпульс при перезапуску.

9. ***Інкрементний лічильник CTU:***



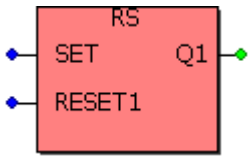
CU	BOOL – вхід для лічіння (лічити, коли CU дорівнює TRUE).
RESET	BOOL – команда скидання (домінанта).
PV	DINT - програмоване максимальне значення.
Q	BOOL - переповнення: TRUE, коли CV>=PV.
CV	DINT - результат лічіння.

10. ***Перемикач із домінантою вмикання SR:***



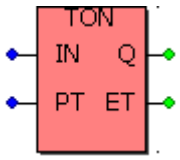
SET1	BOOL - якщо TRUE, встановлює Q1 у TRUE (домінанта).
RESET	BOOL - якщо TRUE, скидає Q1 у FALS.
Q1	BOOL - логічний стан пам’яті.

11. ***Перемикач із домінантою вимкнення RS:***



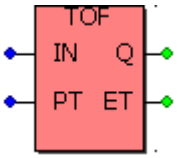
SET BOOL - якщо TRUE встановлює Q1 у TRUE .
RESET1 BOOL - якщо TRUE, скидає Q1 у FALS (домінанта).
Q1 BOOL - логічний стан пам'яті.

12. ***Таймер із затримкою вмикання TON***



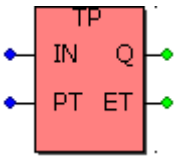
IN BOOL - якщо передній фронт, то починає збільшувати внутрішній таймер, якщо задній фронт, то зупиняється і скидає внутрішній таймер.
PT TIME - максимальний програмований час.
Q BOOL - якщо TRUE, програмований час використаний.
ET TIME - поточний використаний час.

13. ***Таймер із затримкою вимикання TOF***



IN BOOL - якщо задній фронт, то починає збільшувати внутрішній таймер, якщо передній фронт, то зупиняється і скидає внутрішній таймер.
PT TIME - максимальний програмований час.
Q BOOL - якщо TRUE, програмований час використаний.
ET TIME - поточний використаний час.

14. ***Час пульсування TP***



IN BOOL - якщо передній фронт, то починає збільшувати внутрішній таймер (якщо вже не збільшується).Якщо FALSE і лише, якщо таймерний інтервал вичерпано, скидає внутрішній таймер. Будь-яка зміна IN під час лічіння не має ефекту.
PT TIME - максимальний програмований час.
Q BOOL - якщо TRUE таймер лічить.
ET TIME - поточний використаний час.

SFC- мова.

15. ***Крок*** у SFC- програмі кроки зображується у вигляді прямокутника. До кожного кроку праворуч приєднані прямокутники дій. У лівій частині прямокутника дій знаходиться класифікатор, можливо, з константою часу. Класифікатор визначає спосіб впливу активного кроку на дію.

Класифікатори SFC-мови

Класифікатор	Дія	Коментар
N (Non-stored)	Що не зберігається	Дія виконується у кожному робочому циклі протягом активності кроку
R (Reset)	Позачерговий скид	Дія деактивується
S (Stored)	Що зберігається	Дія активна доки немає скиду. Дія продовжує виконання у кожному циклі, навіть тоді, коли крок вже не активний
L (timeLimited)	Що обмежена за часом	Дія активна протягом вказаного часу, але не триваліше часу активності кроку
D (Delayed)	Що відкладена	Дія активується через заданий час після активації кроку і залишається бути активною, доки крок є активним
P (Pulse)	Імпульс	Дія виконується один раз при активації і другий раз після деактивації кроку
SD (Storedand time Delayed)	Що зберігається і відкладена	Дія активується через заданий час після активації кроку, навіть якщо крок вже неактивний
DS (Delayed and Stored)	Що відкладена і зберігається	Дія активується через заданий час після активації кроку і залишається активною до скиду
SL (Stored and Limited)	Що зберігається і обмежена за часом	Дія активується разом з кроком і залишається бути активною заданий час незалежно від активності кроку

Класифікатори L, D, SD, DS і SL потребують установлення константи у форматі TIME.

16. ***Переходи*** зображуються маленькими горизонтальними смужками, що перетинають лінії сполучення. Перехід виконується, коли є дозвіл (відповідний йому крок активний) и при цьому умова переходу має значення TRUE.