МІНІСТЕРСТВО ОСВІТІ І НАУКИ УКРАЇНІ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНІЙ УНІВЕРСИТЕТ
імені ВОЛОДИМИРА ДАЛЯ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

«ТЕХНОЛОГІЧНЕ ПРОГРАМУВАННЯ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМ УПРАВЛІННЯ»

(*для здобувачів вищої освіти денної та заочної форм навчання спеціальності 151 Автоматизація та комп'ютерно–інтегровані технології*)

(*Електронне видання*)

З А Т В Е Р Д Ж Е Н О
на засіданні кафедри комп'ютерно-інтегрованих систем управління
Протокол № 4 від 02.12.2021 р.

СЄВЄРОДОНЕЦЬК 2021

УДК 681.518

Методичні вказівки до виконання лабораторних робіт з дисципліни «Технологічне програмування комп’ютерно-інтегрованих систем управління» (для здобувачів вищої освіти денної та заочної форм навчання спеціальності 151 Автоматизація та комп’ютерно–інтегровані технології) / Укладачі: О.В. Кузнецова, О.І. Проказа. – Сєвєродонецьк: Вид-во СНУ ім. В. Даля, 2021. – 96 с.

Наведені методичні вказівки до виконання лабораторних робіт з дисципліни «Технологічне програмування комп’ютерно-інтегрованих систем управління» для здобувачів вищої освіти денної та заочної форм навчання. Описані етапи створення додатків в середовищі ISaGRAF на мовах технологічного програмування, постановка завдань, варіанти завдань, приклади виконання.

Укладачі: О.В. Кузнецова, ст. викл.

О.І. Проказа, к.т.н., доц.

Рецензент М.Г. Лорія, д.т.н., проф.

ЗМІСТ

ВСТУП

Методичні вказівки складені відповідно до робочої програми з курсу «Технологічне програмування комп’ютерно-інтегрованих систем управління» спеціальності 151 – Автоматизація та комп’ютерно–інтегровані технології.

«Технологічне програмування комп’ютерно-інтегрованих систем управління» – це технічна дисципліна, в якій вивчають мови технологічного програмування для рішення задач управління технологічними процесами.

Мета дисципліни «Технологічне програмування комп’ютерно-інтегрованих систем управління» – ознайомлення здобувачів вищої освіти з сучасними системами технологічного програмування контролерів і системами диспетчеризації.

Завдання дисципліни «Технологічне програмування комп’ютерно-інтегрованих систем управління» – закріплення практичних знань та умінь здобувачів розробляти програмне забезпечення для комп’ютерно-інтегрованих систем управління.

У результаті вивчення дисципліни студент зобов’язаний

знати:

- архітектуру сучасних комп’ютерно-інтегрованих систем управління;
- загальні характеристики, типи, структури, функції та загальні принципи побудови комп’ютерно-інтегрованих систем управління;
- сучасні системи технологічного програмування;

вміти:

- використовувати спеціалізовані середовища для побудови комп'ютерно-інтегрованих систем управління;
- розробляти прикладне програмне забезпечення для комп'ютерно-інтегрованих систем управління;
- розробляти прикладне програмне забезпечення контролерів різних виробників з використанням технологічних мов програмування міжнародних стандартів МЕК 61131.

ЗАГАЛЬНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Порядок виконання лабораторних робіт

Для виконання лабораторних робіт з курсу «Технологічне програмування комп'ютерно-інтегрованих систем управління» необхідно уважно ознайомитися з теоретичними основами кожної роботи, приведеними в методичних вказівках. Для більш глибокого ознайомлення з матеріалом, необхідно звертатися до конспекту лекцій, а також до технічної літератури, на яку приводяться посилання у вказівках.

Якщо під час вивчення дисципліни з'являться запитання, на які не знаходиться відповідь у конспекті лекцій або технічній літературі, необхідно звертатись за консультацією до ведучого лектора.

Лабораторна робота складається з:

- одержання індивідуального завдання;
- одержання шуканих результатів;
- оформлення й захист звіту з лабораторної роботи.

Лабораторні роботи необхідно виконувати за допомогою персонального комп'ютера або ноутбука з використанням системи ISaGRAF.

Правила оформлення звітів з лабораторних робіт

Звіт повинен містити:

- титульний аркуш із найменуванням лабораторної роботи й даними виконавця;
- мета роботи;
- постановка задачі відповідно до варіанту;
- порядок і результати дослідження в числовому і графічному вигляді.
- аналіз результатів і висновки.

ЛАБОРАТОРНА РОБОТА № 1

Тема: ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ ISaGRAF НА МОВІ FBD

Мета роботи: знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування FBD.

Короткі теоретичні відомості

Інструментальна система ISaGRAF

Однією з найпоширеніших інструментальних систем технологічного програмування контролерів, що реалізують стандарт IEC (МЕК) 61131-3, є система ISaGRAF. Інструментальна система ISaGRAF відноситься до класу систем CASE-типу (Computer Aided Software Engineering).

Система ISaGRAF включає в себе систему розробки (ISaGRAF Workbench) і систему виконання (ISaGRAF Target).

Загальна структура системи ISaGRAF представлена на рисунку 1.

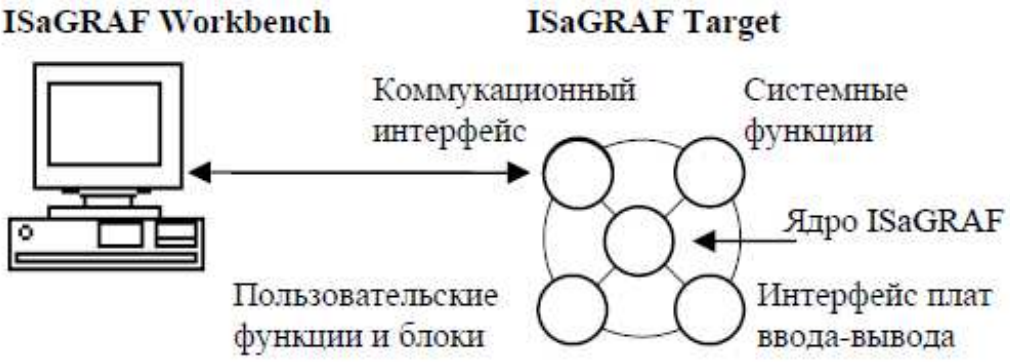


Рисунок 1 – Загальна структура системи ISaGRAF

Система розробки призначена для створення прикладних задач, які виконуються під управлінням ядра ISaGRAF на системах виконання, і встановлюється на IBM PC-сумісний комп'ютер під управлінням операційної

системи сімейства MS Windows. Система розробки компілює проєкт в системно-незалежний код - Target Independent.

Code (TIC). TIC-код завантажується через канали комунікацій (RS-232, RS-485, Ethernet TCP / IP) в цільову машину (контролер) для виконання.

Система виконання завантажується або пропалюється в ПЗУ цільової машини. Вона включає в себе ядро ISaGRAF і набір додаткових модулів.

Ядро ISaGRAF реалізує підтримку стандартних мов програмування ПЛК, набору стандартних функцій і функціональних блоків.

Комунікаційний інтерфейс забезпечує підтримку з боку ПЛК процедури завантаження ISaGRAF-додатка, доступ до змінних цього додатка під час налагодження, а також взаємодія з системами SCADA (Supervisory Control And Data Acquisition - диспетчерське управління і збір даних) за допомогою протоколу Modbus.

Системні функції призначені для опису специфіки апаратної частини і операційної системи, встановленої в конкретному контролері. ISaGRAF-додаток може функціонувати під різними операційними системами (Windows NT, Windows CE, Embedded NT, OS 9/9000, MS DOS, Lynx OS, QNX, VxWorks та ін.).

Технологічні мови програмування контролерів в ISaGRAF

У ISaGRAF закладена методологія структурного програмування, що дозволяє користувачеві представити автоматизований процес в найбільш легкій і зрозумілій формі. Стандартом МЕК 61131-3 визначається п'ять мов: три графічних (SFC, FBD, LD) і дві текстових (ST, IL). Крім цих мов, ISaGRAF пропонує мову блок-схем (FlowChart). Всі ці мови програмування інтегровані в єдину інструментальну середу і працюють з єдиними об'єктами даних.

Проєкт ISaGRAF розділений на кілька програмних модулів, які називаються програмами. Програми проєкту пов'язані в деревоподібну структуру.

Програма - це логічна програмована одиниця, яка описує операції з змінними процесу. Програма описує або послідовні, або циклічні операції.

Циклічні програми виконуються на кожному циклі цільової системи. Виконання послідовних програм визначається динамічними правилами мови SFC.

Програми пов'язані одна з одною в ієрархічне дерево. Програми, що знаходяться нагорі ієрархії, активізуються системою. Підпрограми (нижній рівень ієрархії) активізуються їх батьками.

Одна і та ж програма не може змішувати декілька мов, за винятком LD і FBD, які можуть бути скомбіновані в одній діаграмі.

Будь яка константа, змінна або вираз, що використовуються в програмі, повинні характеризуватися своїм типом. Типи повинні бути узгоджені в графічних операціях і текстових виразах. Ось основні типи програмних об'єктів:

- BOOLEAN – логічна величина;
- ANALOG – ціла або дійсна безперервна величина;
- TIMER – часова величина;
- MESSAGE – рядок символів.

Основний принцип ISaGRAF: СИНХРОНІЗАЦІЯ

Прикладна задача ISaGRAF працює в синхронному режимі за тимчасовими циклами, тривалість яких визначається розробником. Мінімальна тривалість циклів виконання прикладної задачі визначається характеристиками апаратно-програмної платформи (ISaGRAF Target), на якій відбувається виконання завдання.

Програмні одиниці ISaGRAF-проєкту (програми, функції, функціональні блоки) розміщуються в послідовних або циклічних секціях. При цьому програми, розташовані в циклічних секціях виконуються повністю в кожному ISaGRAF-циклі. Програмний цикл розпочинається опитуванням всіх сконфігурованих зовнішніх каналів датчиків (наприклад, канали АЦП) і завершується відновленням всіх вихідних каналів (напр. канали ЦАП). Така схема роботи ISaGRAF-додатка гарантує користувачеві, що в рамках одного тимчасового циклу він буде працювати тільки з однією копією об'єктних даних типу INPUT/OUTPUT.

Етапи створення додатка в середовищі ISaGRAF

У загальному вигляді роботи з створення додатків у середовищі ISaGRAF можуть бути розбиті на наступні етапи:

1. Створення проєкту.
2. Створення програм.
3. Оголошення змінних.
4. Редагування програми.
5. Конфігурація введення/виведення.
6. Установка опцій додатка і параметрів зв'язку з контролером.
7. Компіляція програм і створення коду додатка.
8. Симуляція та налагодження додатка.
9. Загрузка додатка до контролера.

МОВА FBD

Мова функціональних блочних діаграм (Functional Block Diagram, FBD) – це графічна мова. Найбільш підходить для управління безперервними процесами та регулювання. FBD-програма побудована з стандартних елементарних функціональних блоків із бібліотеки ISaGRAF: арифметичних, тригонометричних, регуляторів, мультиплексорів і т.д.

Основний формат діаграм FBD

FBD-діаграма описує функцію між **вхідними** та **вихідними змінними** (рисунок 2). Функція описується як безліч елементарних блоків. Вхідні і вихідні змінні приєднуються до блоків за допомогою ліній зв'язку. Вихід блоку може бути також з'єднаний з входом іншого блоку.

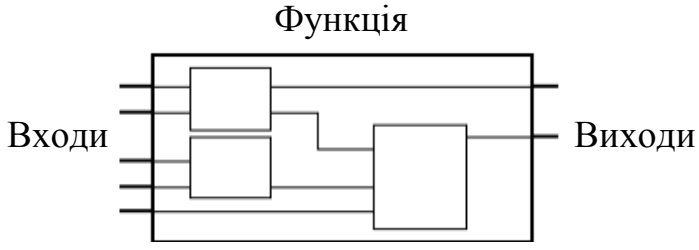


Рисунок 2 – Структура FBD-діаграми

Повна функція, яка використовується FBD програмою, будується на основі стандартних елементарних блоків з стандартної системної бібліотеки або з секцій ресурсу 'Функції' і 'Функціональні блоки'. Кожен блок має фіксовану кількість вхідних точок підключення і фіксовану кількість вихідних точок підключення. Блок представляється одиночним прямокутником (рисунок 3). Входи з'єднуються з його лівим краєм. Виходи з'єднуються з його правим краєм. Елементарний блок реалізує одну функцію між його входами і виходами. Ім'я функції, яка реалізується блоком, пишеться в його символі прямокутника. Кожен вхід або вихід блоку мають чітко визначений тип.

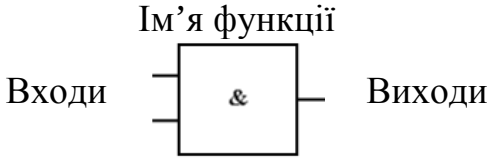


Рисунок 3 – Структура функціонального блоку

Вхідні змінні FBD програми повинні бути з'єднані з точками входу блоку. Тип кожної змінної повинен бути тим же, що і тип відповідного входу. *Входом FBD блоку може бути константа, будь-яка внутрішня, вхідна або вихідна змінна*.

Вихідні змінні FBD програми повинні бути з'єднані з точками виходу блоку. Тип кожної змінної повинен бути тим же, що і тип відповідного виходу. *Виходом FBD блоку може бути будь-яка внутрішня або вихідна змінна, або ім'я функції (тільки функція).* Коли вихід є ім'ям поточної редагованої функції, він представляє привласнення значення для функції, що повертається (що повертається в програму, яка викликає).

Вхідні та вихідні змінні, входи і виходи функціональних блоків з'єднані лініями або **зв'язками**. Поодинокі лінії можуть бути використані для з'єднання двох логічних точок діаграми: вхідної змінної і входу функціонального блоку; виходу функціонального блоку і входу іншого блоку; виходу функціонального блоку і вихідної змінної (рисунок 4).

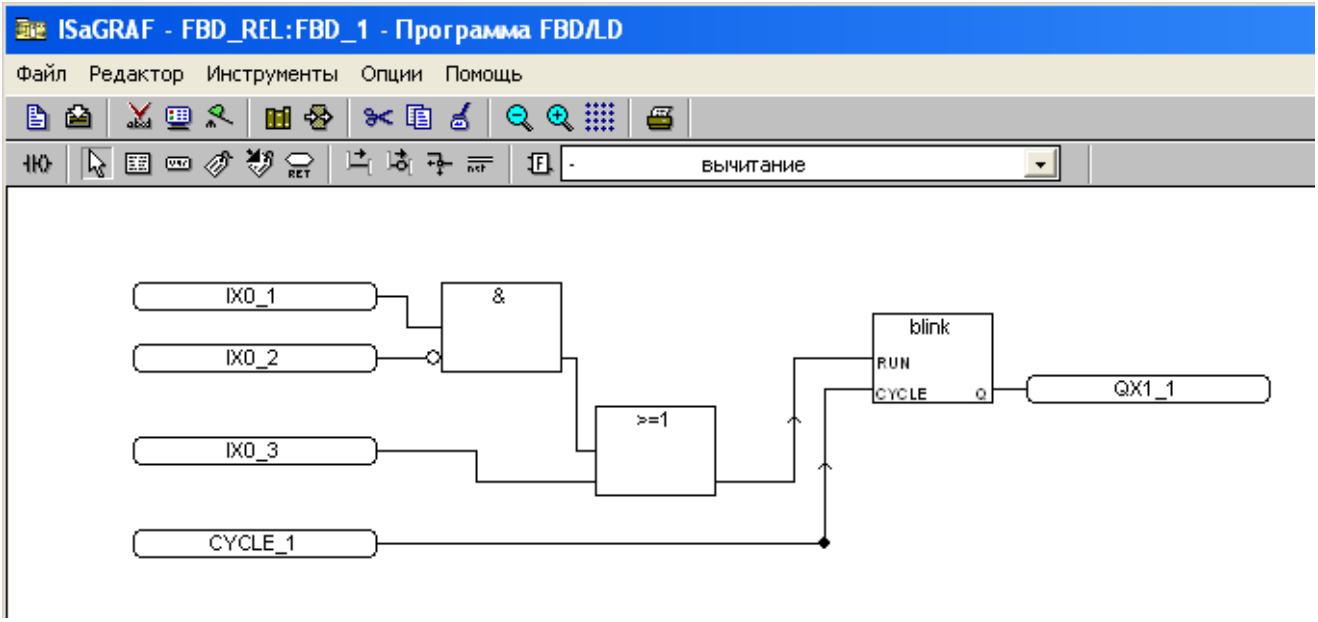


Рисунок 4 – Приклад програми мовою FBD

Зв’язок орієнтований, це означає, що дані передаються з лівого кінця до правого. Лівий та правий кінці зв’язку повинні бути **одного типу**.

Множинне праве з'єднання, назване також **розбіжністю**, може бути використано для передачі інформації від лівого кінця до кожного правого кінця. Всі кінці з'єднання повинні бути одного типу.

Використання оператора RETURN

Ключове слово **RETURN** може бути виходом діаграми. Воно повинно бути зв’язано з логічним виходом функціонального блока. Оператор **RETURN** представляє собою **умовне завершення** програми: якщо вихід блока, зв’язаного з оператором, має тип **TRUE**, решта діаграми не виконується.

Приклад використання оператора RETURN приведений на рисунку 5.

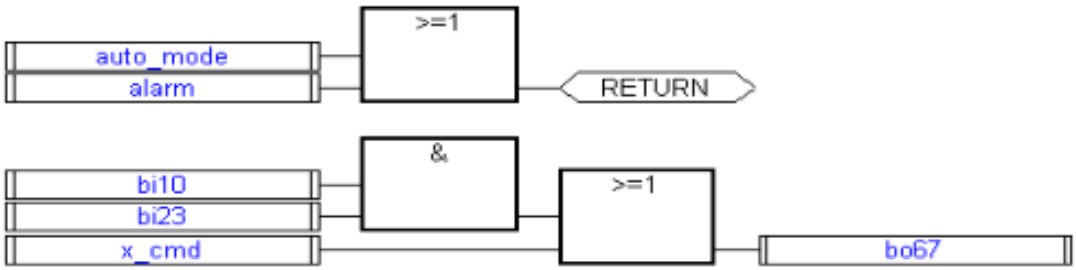


Рисунок 5 – Приклад використання оператора RETURN

Використання стрибків та міток

Стрибки і мітки використовуються для керування виконанням діаграми. До правого краю символу мітки або стрибка не може бути приєднано ніяких інших об’єктів. Використовуються наступні позначення:

>>LAB..... стрибок на мітку (ім’я мітки "LAB")

LAB:..... визначення мітки (ім’я мітки "LAB")

Якщо лінія зв’язку зліва від символу стрибка знаходиться в стані TRUE, то виконання програми переходить на відповідну мітку.

Приклад використання стрибка та мітки приведений на рисунку 6.

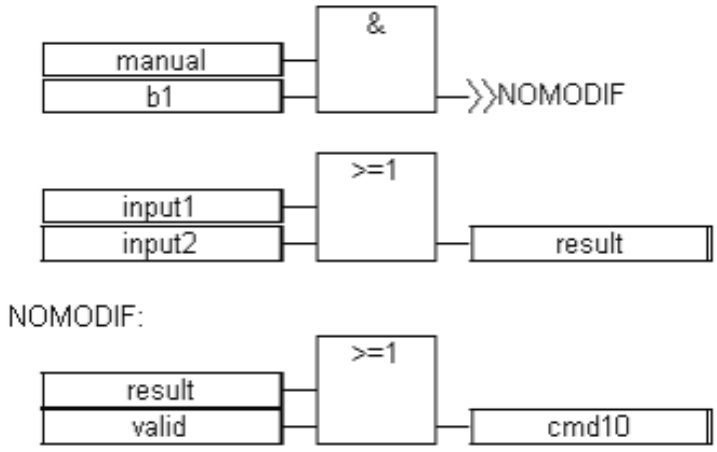


Рисунок 6 – Приклад використання стрибка та мітки

Логічне заперечення

Одиночна лінія зв'язку з правим кінцем, приєднаним до входу функціонального блоку, може закінчуватися логічним запереченням. Заперечення є маленьким колом. Коли використовується логічне заперечення, лівий і правий кінці лінії зв'язку повинні мати тип **BOOLEAN.**

Приклад використання заперечення приведений на рисунку 7.

Рисунок 7 – Приклад використання логічного заперечення

Виклик функцій і функціональних блоків

Мова FBD дозволяє викликати функції або функціональні блоки. Функція або функціональний блок представляються прямокутником. Ім'я, написане в прямокутнику, є ім'ям функції або функціонального блоку.

У разі функції єдиним виходом прямокутника є значення, що повертається. Функціональні блоки можуть мати кілька виходів.

Приклад використання виклику функції приведений на рисунку 8.

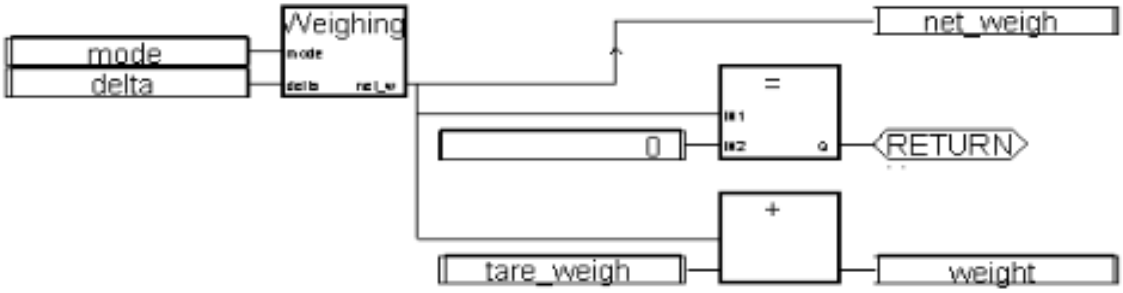
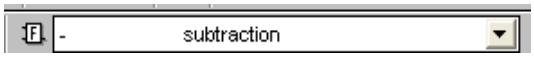


Рисунок 8 – Приклад використання виклику функції

Для введення функціональних блоків використовується вікно 

панелі інструментів, де можна вибрати блоки операторів і функцій мови FBD (рисунки 9 – 10).

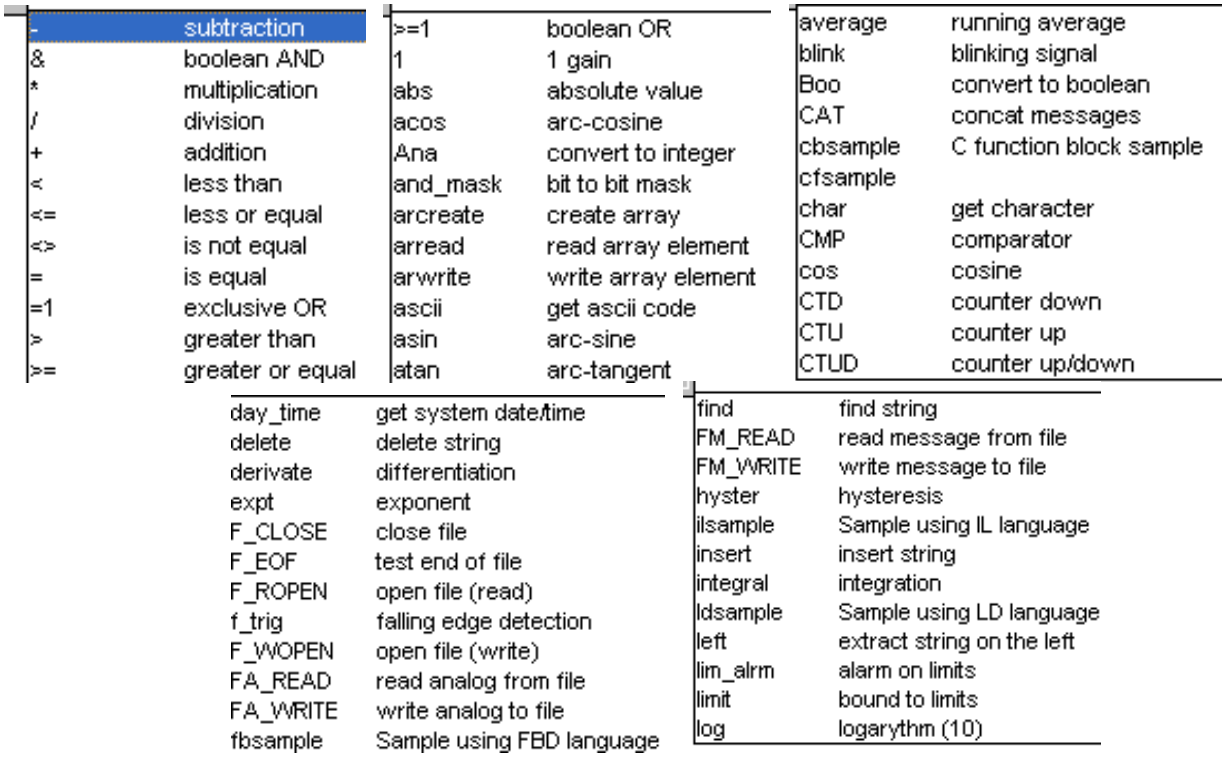


Рисунок 9 – Фрагменти вікон з операторами і функціями мови FBD

max	maximum value
mid	extract string
min	minimum value
mlen	get string length
mod	modulo
Msg	convert to message
mux4	multiplexer 4 inputs
mux8	multiplexer 8 inputs
Neg	numerical negation
not_mask	bit to bit mask
odd	odd parity test
operate	

or_mask	bit to bit mask
pow	power
r_trig	rising edge detection
rand	random value
Real	convert to real
replace	replace string
right	extract string on the right
rol	rotate left
ror	rotate right
RS	reset dominant bistable
sel	binary selector
sema	semaphore

sin	sine
sqrt	square root
SR	set dominant bistable
stackint	stack of integers
stsample	Sample using ST language
system	
tan	tangent
Tmr	convert to timer
TOF	off timer
TON	on timer
TP	pulse timer
trunc	truncate real

Рисунок 10 – Фрагменти вікон з операторами і функціями мови FBD

Приклад виконання роботи

Постановка задачі. Розробити та дослідити додаток на мові FBD для віртуального контролера, який реалізує обчислення наступних арифметичних і логічних виразів:

$$y(x_1, x_2, x_3) = \sin\frac{x_3 x_1^{\,2}}{x_3 + x_2},$$

$$f(z_1, z_2, z_3) = \overline{\overline{z}_1 \cap z_2} \cup z_3,$$

де x_1, x_2, x_3 – вхідні дійсні змінні; y – вихідна дійсна змінна; z_1, z_2, z_3 – вхідні логічні змінні; f – вихідна логічна змінна.

Рішення задачі

Загрузка системи ISaGRAF. Натисненням кнопки Пуск → Програми → ISaGRAF 3.4 → Projects виконується загрузка системи ISaGRAF.

Створення проєкту. У вікні Project Management необхідно створити проєкт під назвою "project1" за допомогою команди "New" з меню "File" або кнопки (Create new project) (рисунок 11).

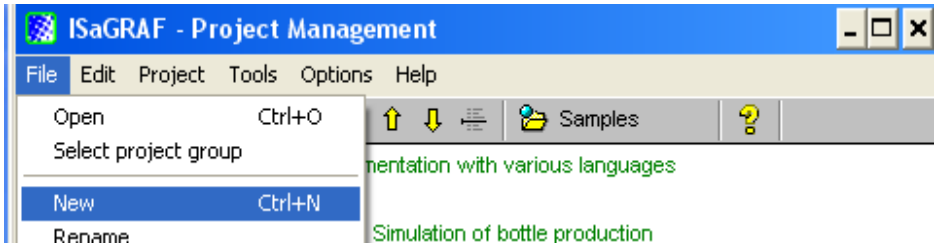


Рисунок 11 – Вибір команди для створення проєкту

У діалоговому вікні "Create new project" (рисунок 12):

• Введіть ім’я проєкту "project1".

• Виберіть конфігурацію введення/виведення [none].

• Натисніть кнопку "Ok".

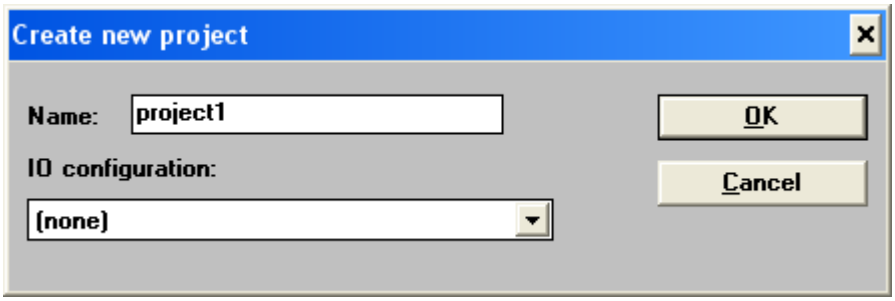


Рисунок 12 – Вікно створення нового проєкту

Відкриття проєкту. Програми проєкту з’являються при відкритті вікна менеджера програм ISaGRAF Programs. Для переходу до вікна менеджера програм клацніть два рази мишею на імені потрібного проєкту ("project1") або використайтеся кнопкою (Open) вікна Project Management.

Створення програм. Вікно Programs зараз відкрито і пусте (тому що жодна програма не визначена) (рисунок 13).

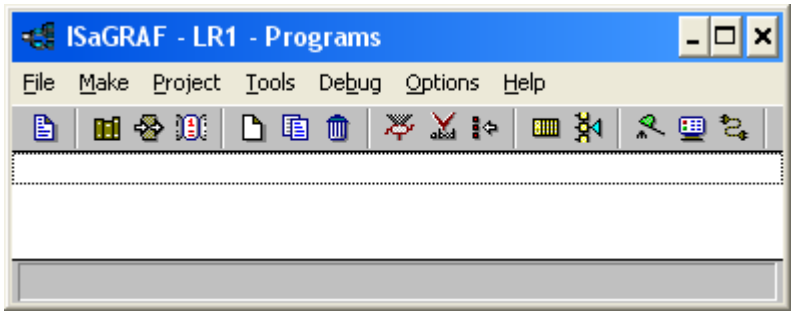


Рисунок 13 – Вікно менеджера програм

Перша програма створюється за допомогою команди "**New**" з меню "**File**" або кнопкою (**Create new program**).

У вікні діалога "**New Program**":

• Уведіть ім’я програми "program1".

• Виберіть мову "**FBD**".

• Виберіть стиль "Begin: Main program (Основна програма)".

• Натисніть кнопку "Ok" для створення програми (рисунок 14).

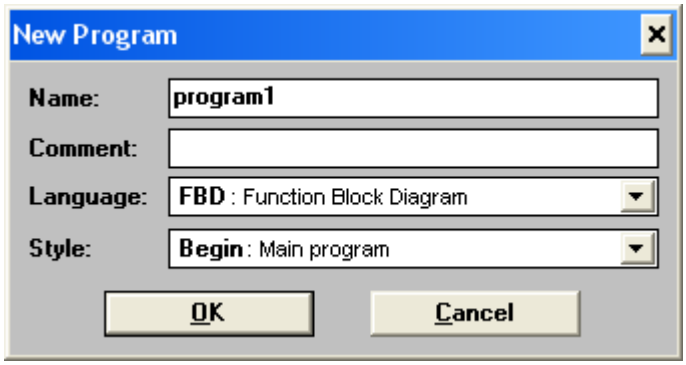


Рисунок 14 – Вікно створення нової програми

Оголошення змінних. Перед введенням програми повинні бути оголошені змінні, використовувані в даній програмі. Це робиться за допомогою команди "**Dictionary (Словник)**" меню "**File**" або кнопки .

Діалогове вікно "**Dictionary**" має кілька закладок: "Booleans (Булеві)", "Integers/Reals (Цілі/Дійсні)", "Timers (Таймери)", "Messages (Повідомлення)", "FB instances (екземпляри)", "Defined words (Макровизначення)", в яких відповідно описуються: булеві, цілі і дійсні, таймерні змінні, повідомлення, екземпляри функціональних блоків, макровизначення (рисунок 15).

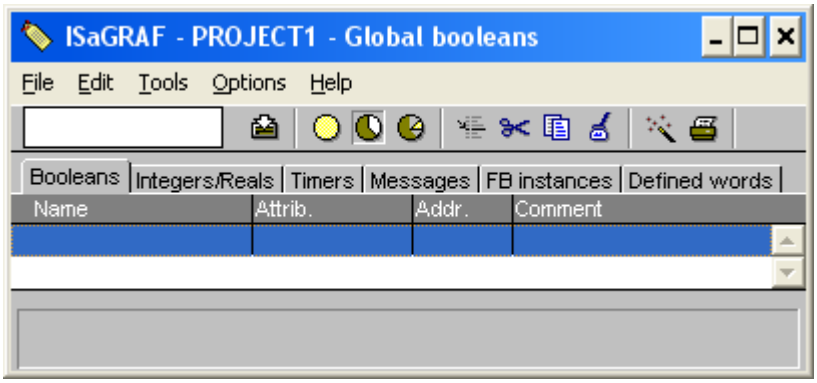


Рисунок 15 – Закладки діалогового вікна "Dictionary"

Створіть всі вхідні і вихідні аналогові (дійсні та цілі) і булеві змінні.

Виберіть закладку **Integers/Reals**, створіть аналогові змінні x1 – x3 (input), y (output), як показано на рисунках 16 – 17.

Перейдіть на вкладку **Booleans**, аналогічно створіть булеві змінні z1 – z3 (input), f (output), як показано на рисунках 18 – 19.

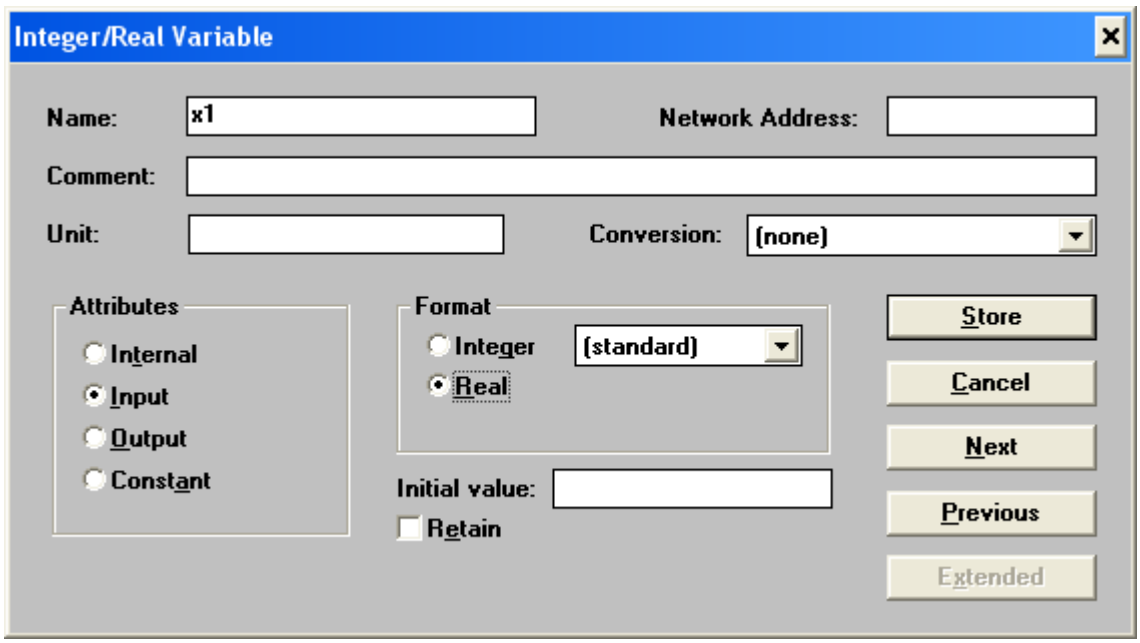


Рисунок 16 – Закладка Integers/Reals

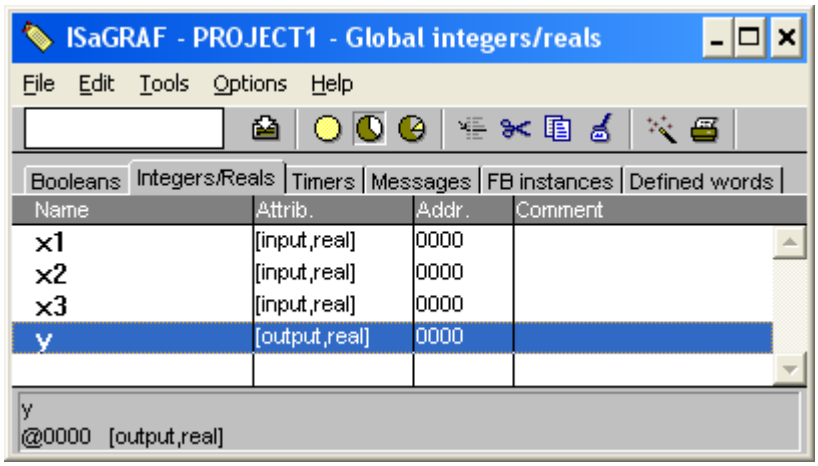


Рисунок 17 – Створені аналогові змінні

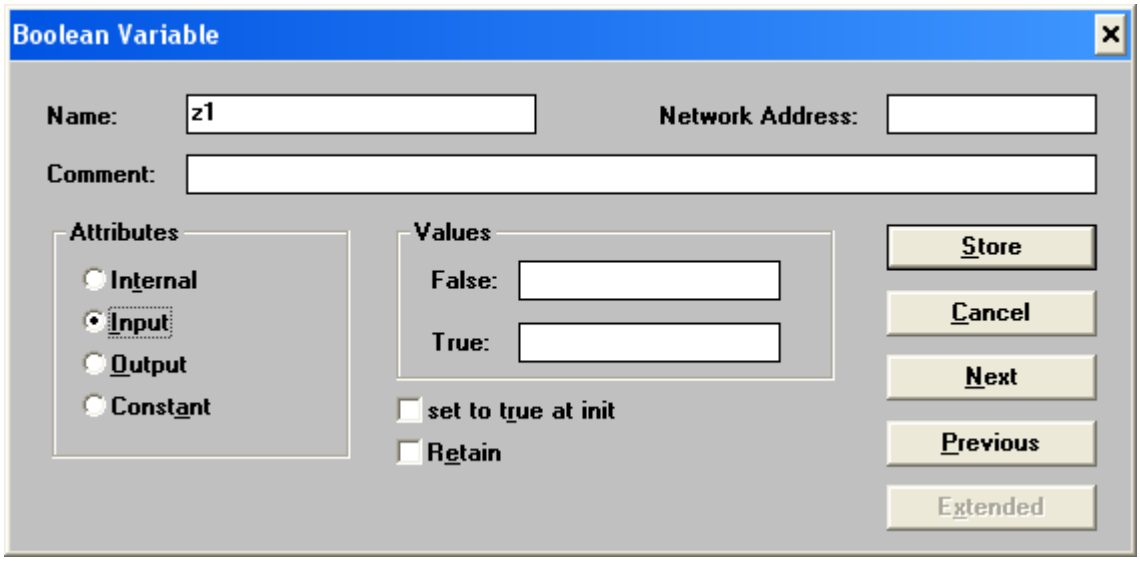


Рисунок 18 – Закладка Booleans

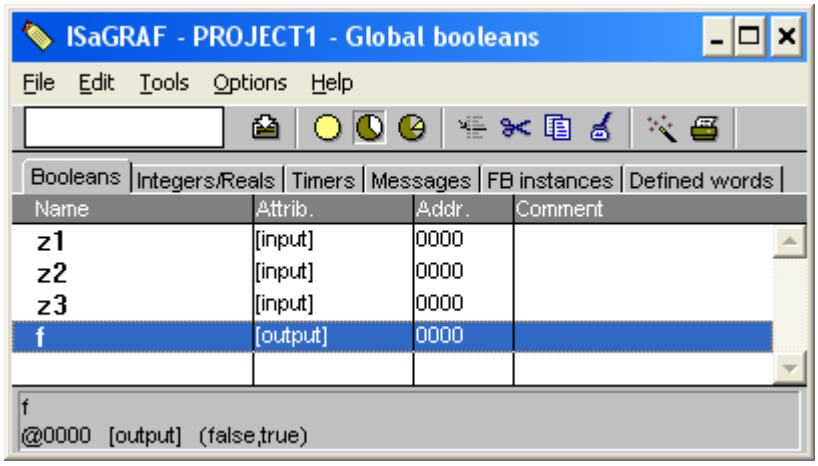


Рисунок 19 – Створені булеві змінні

Наприкінці, покиньте редактор словника, зберігши зміни.

Редагування програм. Команда вікна **Programs** дозволяє змінити зміст програми (рисунок 20).

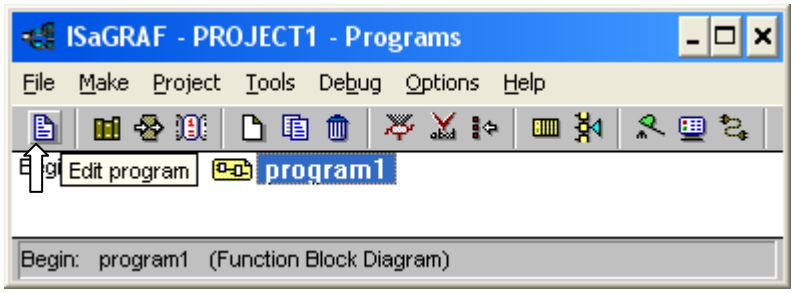


Рисунок 20 – Запуск редагування програми

Використовуваний редактор залежить від мови, вибраної для написання програми.

У вікні редагування програми виконайте набір програми у відповідності з варіантом завдання (рисунок 21).

***Прив'язка змінних введеня/виведення* (*конфігурація введеня/виведення*).** Команда "**I/O connection**" в меню "**Project**" вікна **Programs** або кнопка запускає редактор з'єднання змінних ISaGRAF (рисунок 22). Цей інструмент використовується для створення зв'язків між визначеними в словарі проєкту змінними введення/виведення та відповідною апаратурою.

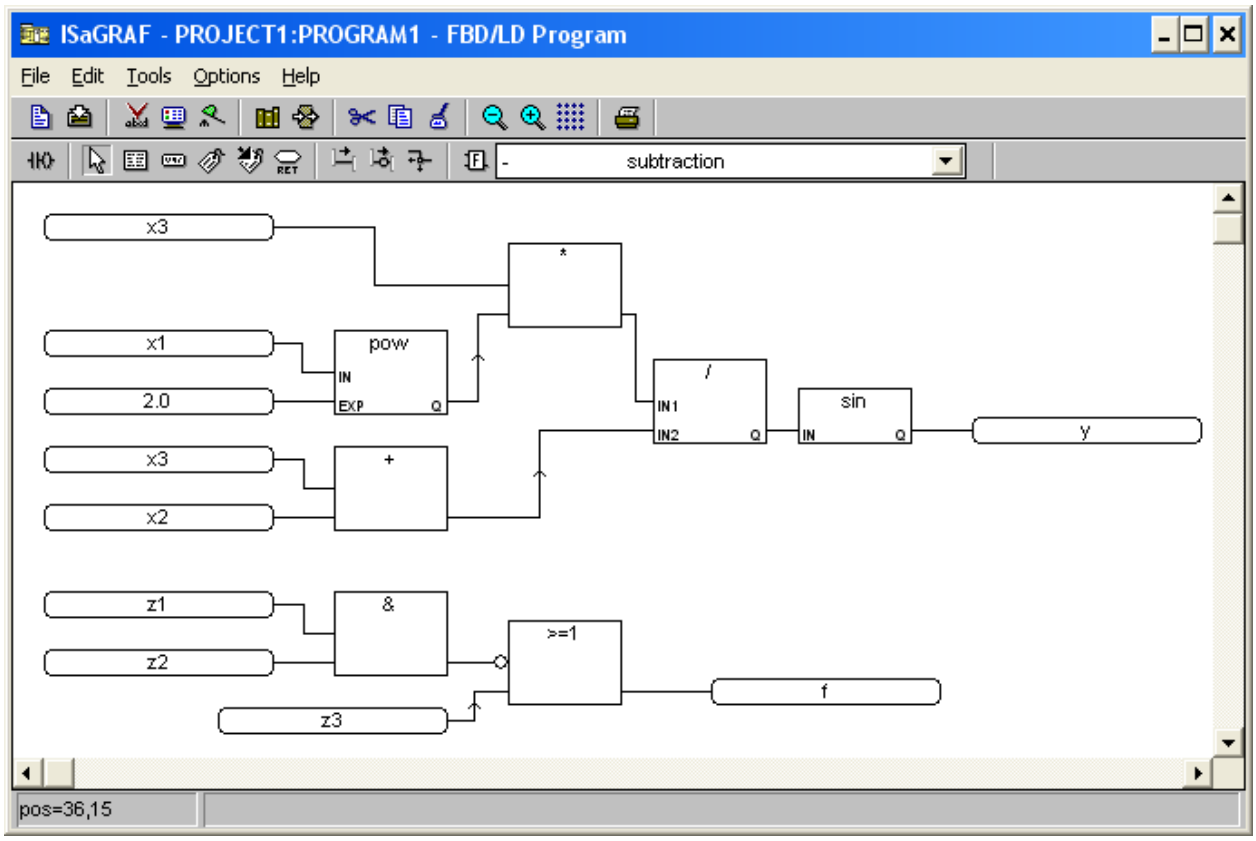


Рисунок 21 – Вікно редагування програми

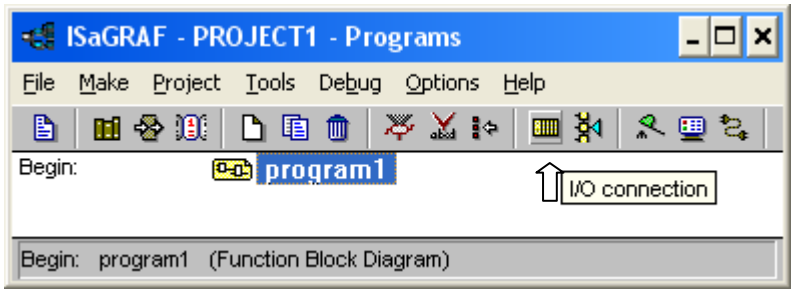


Рисунок 22 – Запуск редактора з’єднання змінних

Для даного завдання виберіть віртуальні плати введення/виведення **xai8**, **xao8**, **xbi8**, **xbo8** (рисунки 23 – 24).

Виконайте прив’язку змінних введення/виведення, визначених в програмі, і входами/виходами плат. Для цього, виділяємо рядок плати **xai8** і в меню "**Edit**" вибираємо команду "**Set channel / parameter**" (рисунок 25).

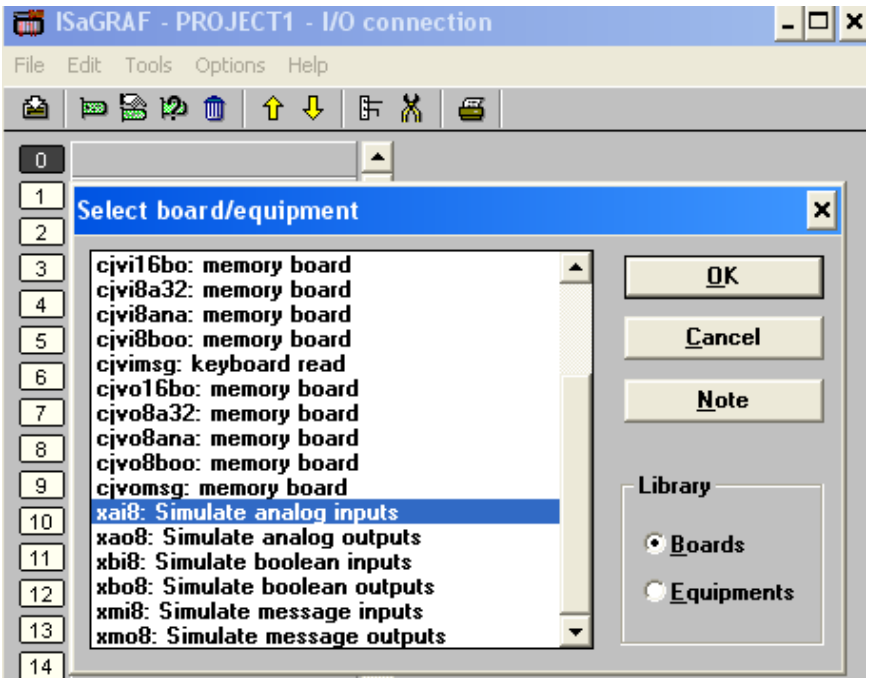


Рисунок 23 – Вибір віртуальних плат введення/виведення

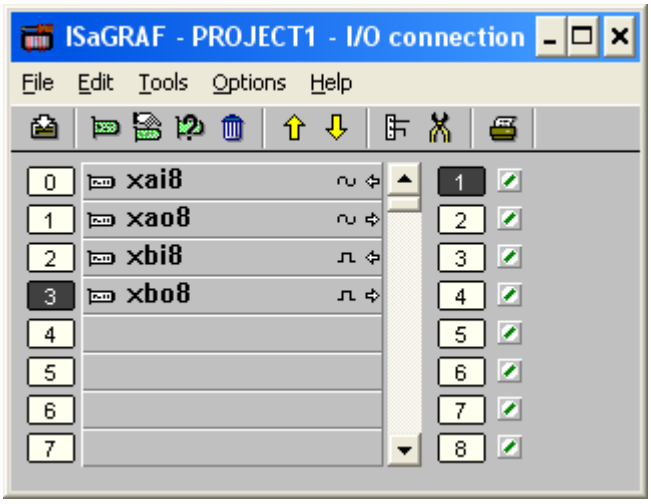


Рисунок 24 – Результат вибору плат введення/виведення

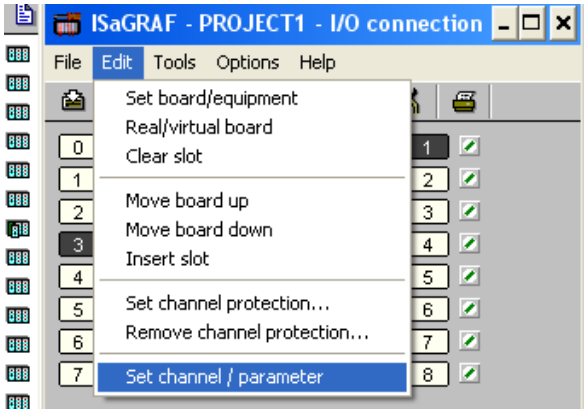


Рисунок 25 – Вибір команди "Set channel / parameter"

З’являється вікно (рисунок 26) з виділеною змінною x1, натиснути кнопку "**Connect**". Аналогічно з’єднайте решту вхідних змінних.

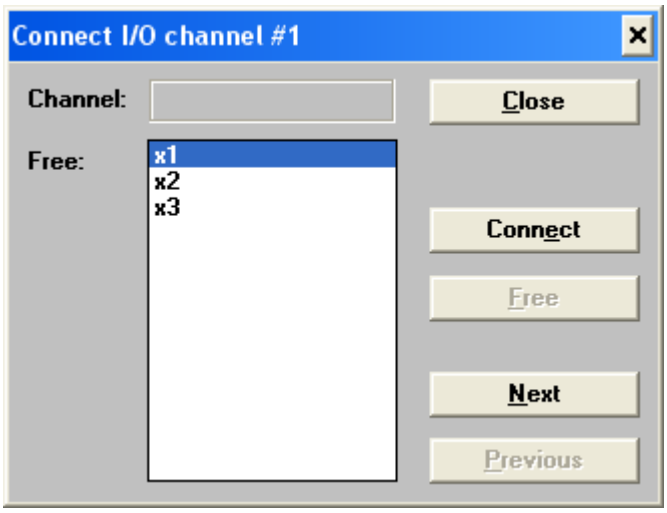


Рисунок 26 – З’єднання змінних з входами плати

Натиснути **Close**. Результат з’єднання показаний на рисунку 27.

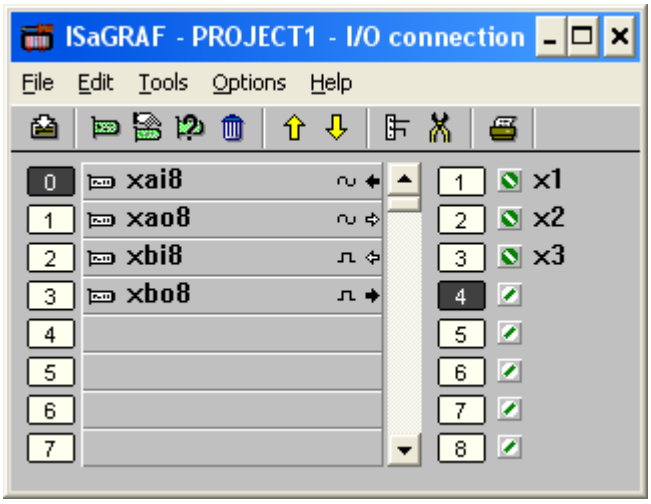


Рисунок 27 – Результат з’єднання вхідних аналогових даних

Результати решти з’єднань приведені на рисунках 28 – 30.

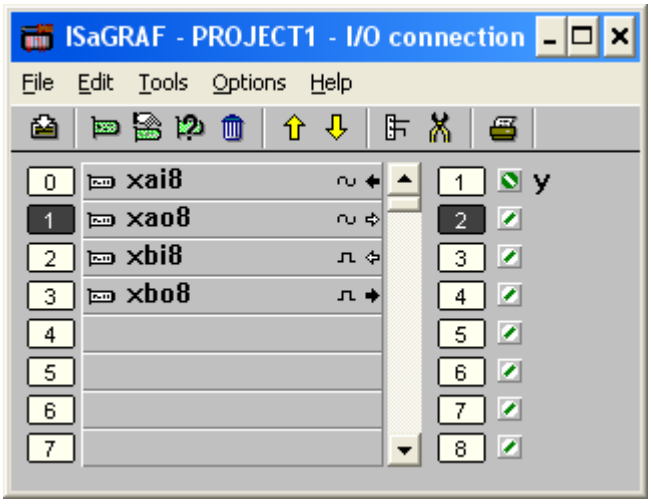


Рисунок 28 – Результат з’єднання вихідних аналогових даних

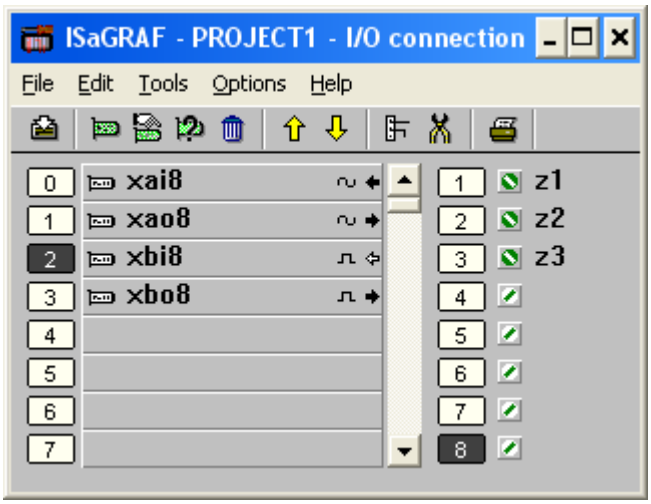


Рисунок 29 – Результат з’єднання вхідних дискретних даних

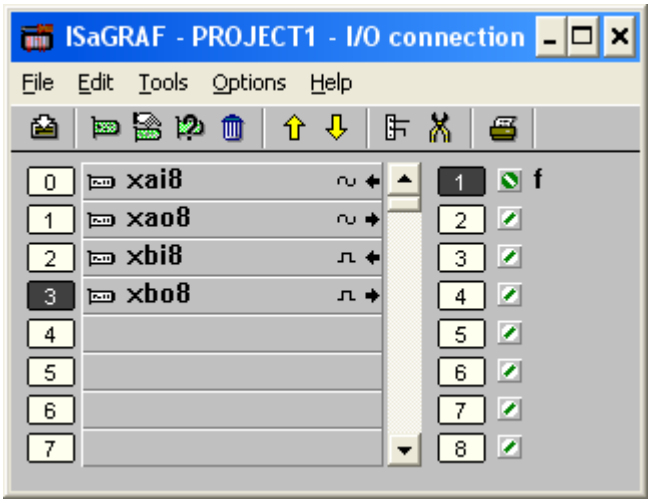


Рисунок 30 –Результат з’єднання вихідних дискретних даних

Закрити вікно ×, з’явиться повідомлення (рисунок 31).

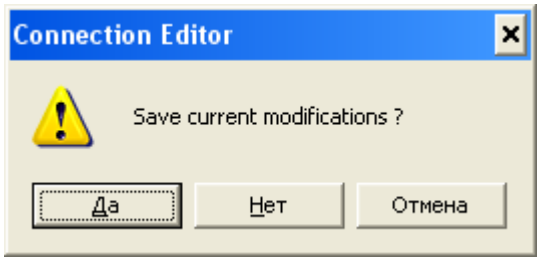


Рисунок 31 –Вікно повідомлення про збереження змін

Підтвердити кнопкою "Да".

Створення коду додатка. Для створення коду використовуйте команду "Make application code" меню "Make" з вікна Programs або кнопку  (Рисунок 32).

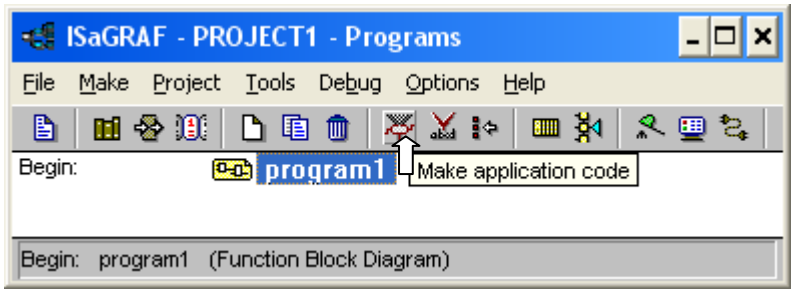


Рисунок 32 – творення коду додатка

Запуститься генератор коду. Якщо помилок немає, то натиснути Exit (рисунок 33).

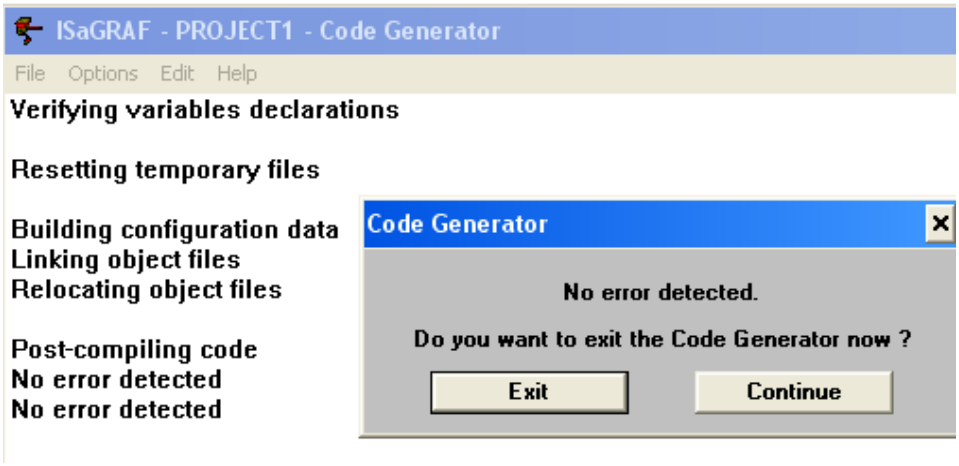


Рисунок 33 – Результат роботи генератора коду

Симуляція. Для запуску ядра симулятора ISaGRAF використовуйте команду "Simulate (Симуляція)" меню "Debug (Налагодження)" з вікна Programs або кнопку  (рисунок 34).

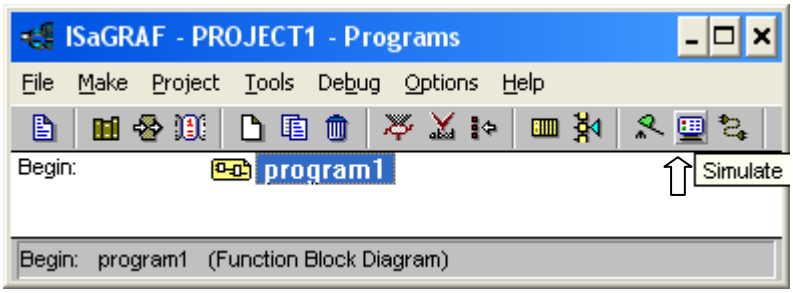


Рисунок 34 – Вікно Programs

Підтвердіть повідомлення (рисунок 35).

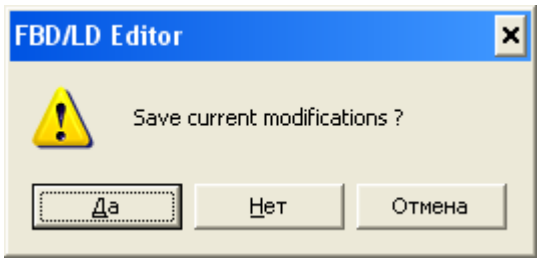


Рисунок 35 – Вікно повідомлення підтвердження

З’явиться вікно симулятора (рисунок 36).

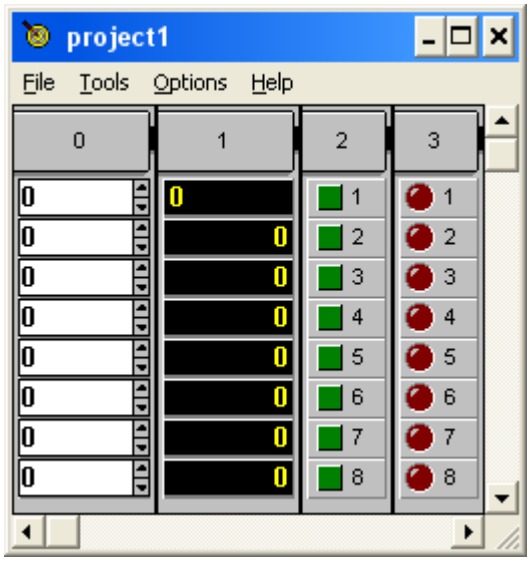


Рисунок 36 – Вікно симулятора

В меню "Options" вікна симулятора відмітити команду "Variable names" (рисунок 37).

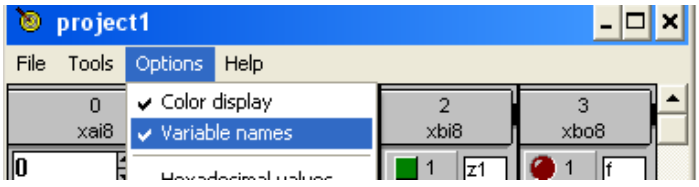


Рисунок 37 – Розвернення імен змінних

Коли з’явиться вікно симулятора, додаток може бути протестовуваний.

Тест № 1

Задати змінним x1, x2, x3 довільні значення. Логічні змінні z1, z2, z3 мають значення false (рисунки 38 – 39).

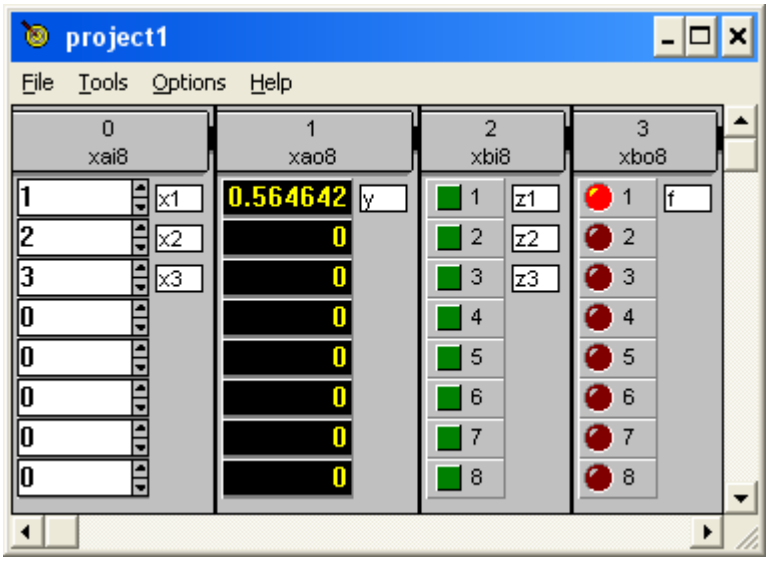


Рисунок 38 – Вікно симулятора з результатом тестування

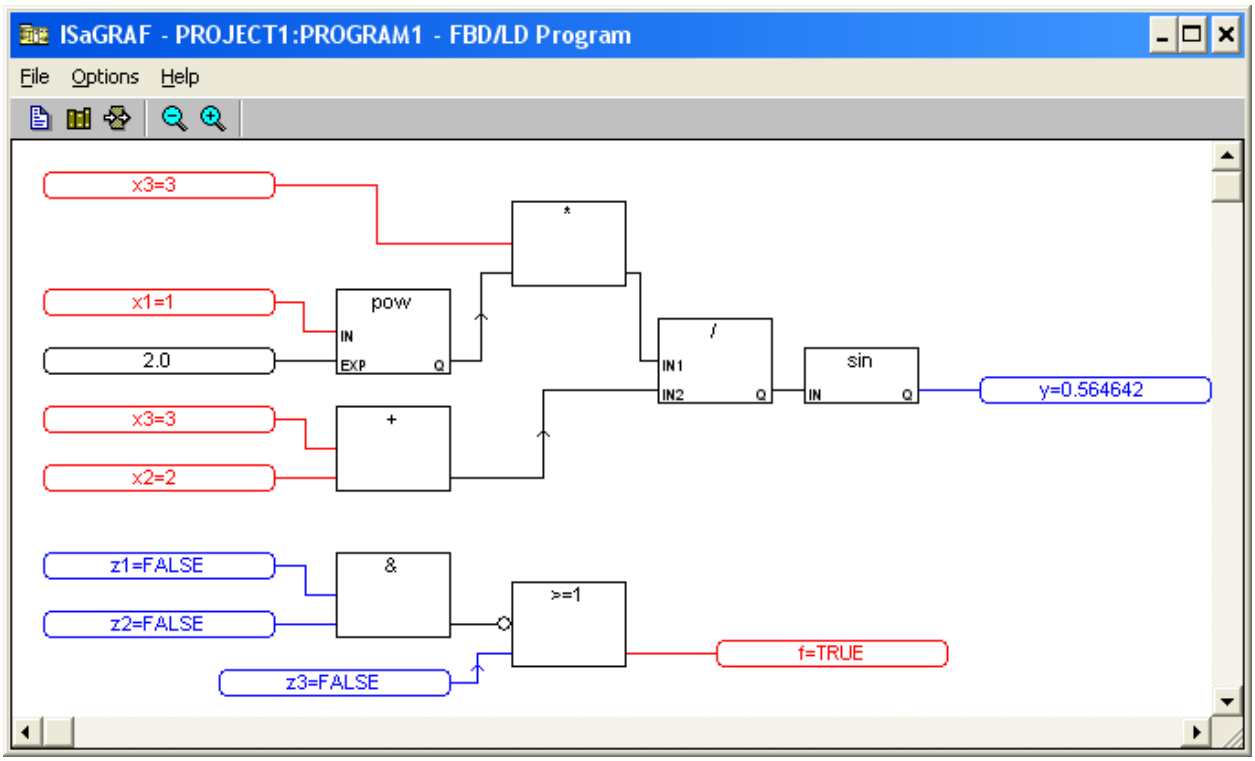


Рисунок 39 – Програма в режимі online

Перевірка результату

$x1 := 1 \qquad x2 := 2 \qquad x3 := 3 \qquad z1 := 0 \qquad z2 := 0 \qquad z3 := 0$

$$y := \sin\left(\frac{x3\,x1^2}{x3 + x2}\right) = 0.564642$$

$$f := \neg(\neg z1 \wedge z2) \vee z3 = 1$$

Тест № 2

Змінити значення змінних x1, x2, x3, а логічні змінні z1, z2 перевести в значення true (рисунки 40 – 41).

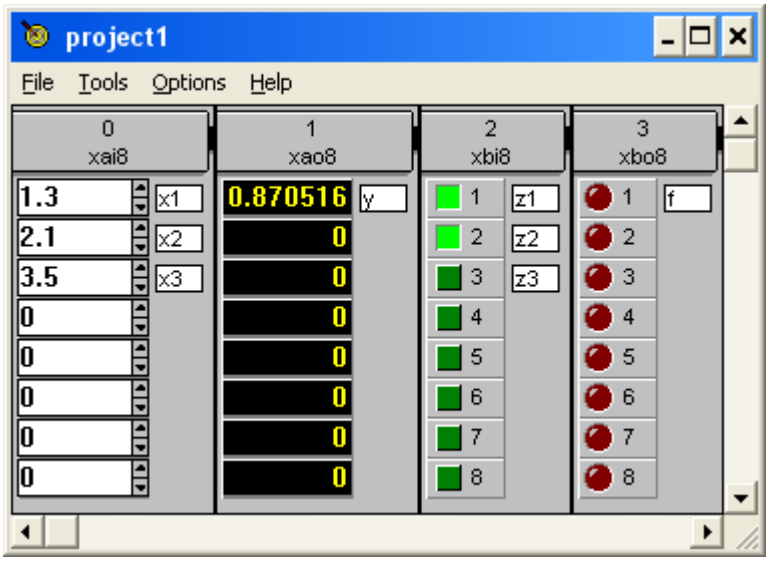


Рисунок 40 – Вікно симулятора з результатом тесту №2

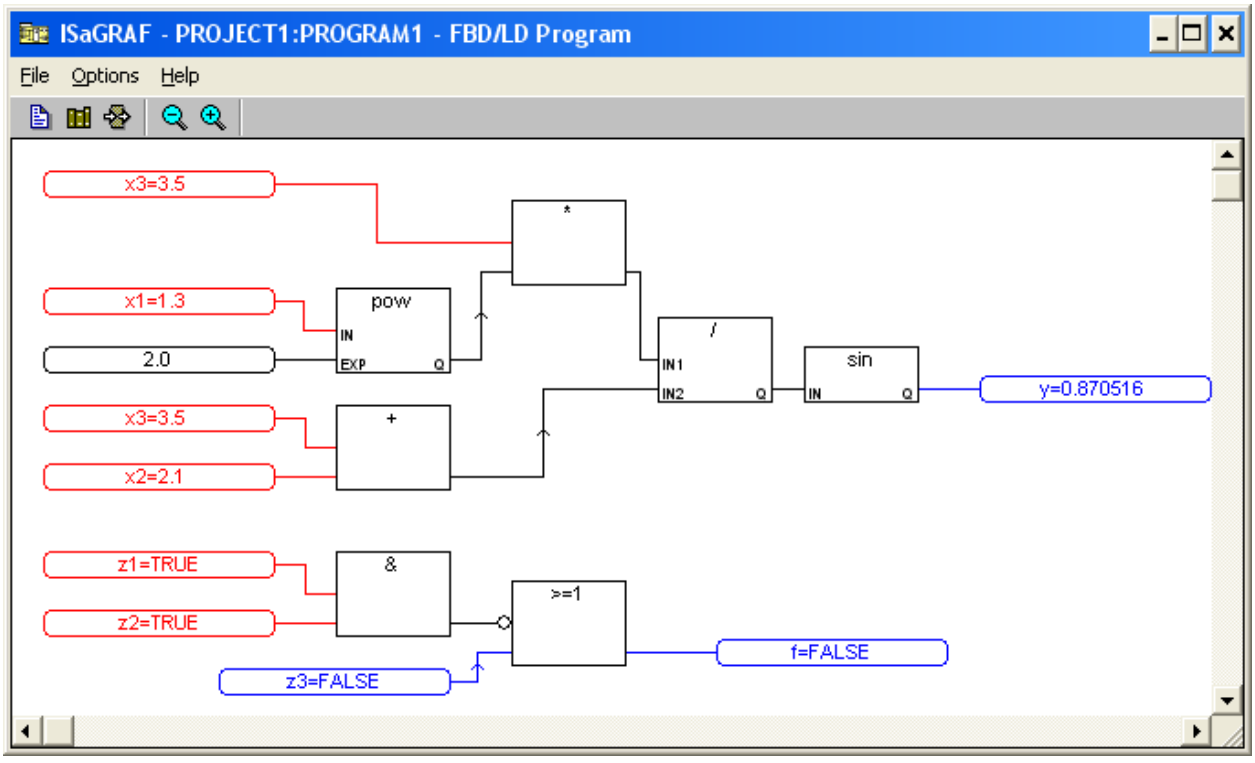


Рисунок 41 – Результат виконання програми

Перевірка результату

$x1 := 1.3 \qquad x2 := 2.1 \qquad x3 := 3.5$

$$y := \sin\left(\frac{x3 \cdot x1^2}{x3 + x2}\right) = 0.870516$$

$z1 := 0 \qquad z2 := 1 \qquad z3 := 0$

$f := \neg(\neg z1 \wedge z2) \vee z3 = 0$

Тест № 3

Ще раз змінити значення вхідних змінних (рисунки 42 – 43).

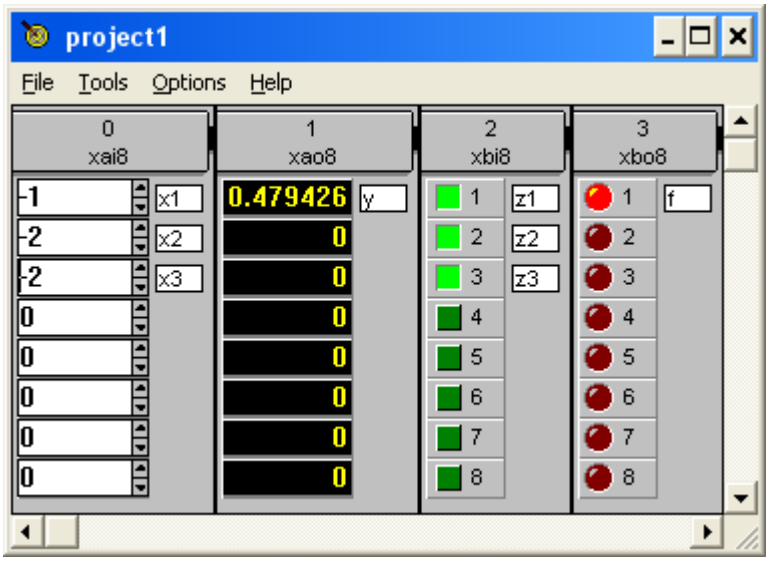


Рисунок 42 – Вікно симулятора з результатом тесту № 3

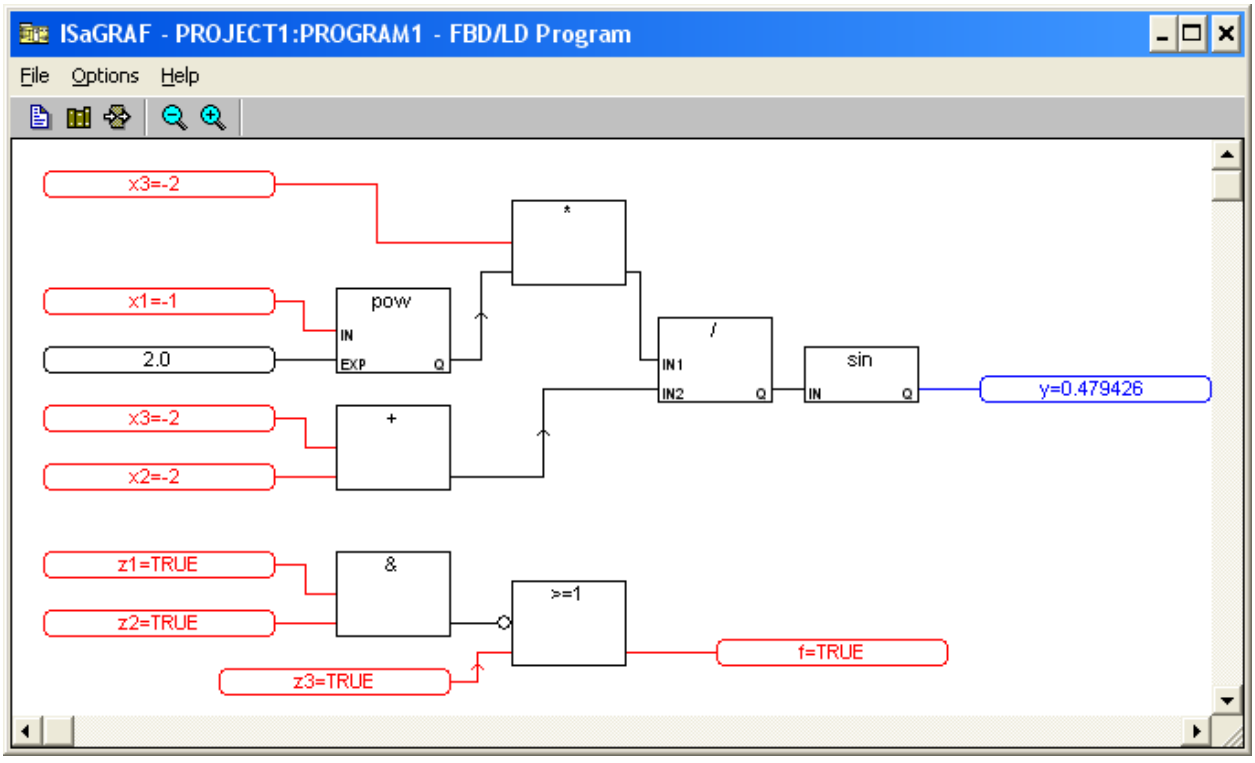


Рисунок 43 – Результат виконання програми

Перевірка результату

$x1 := -1 \qquad x2 := -2 \qquad x3 := -2$

$z1 := 1 \qquad z2 := 1 \qquad z3 := 1$

$$y := \sin\left(\frac{x3\,x1^2}{x3 + x2}\right) = 0.479426$$

$$f := \neg(\neg z1 \wedge z2) \vee z3 = 1$$

Для виходу з симулятора закрийте вікно налагоджувача: меню "file" команда "Exit" або кнопка ✕ (рисунок 44).

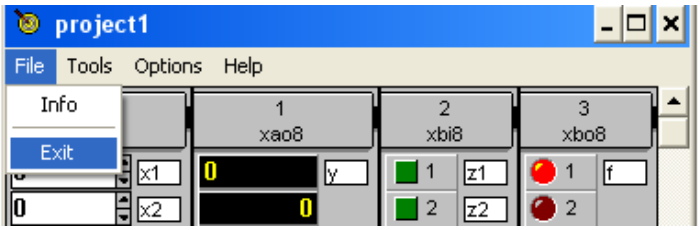


Рисунок 44 – Вихід з симулятору

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№	Завдання 1	Завдання 2
1	$y(x_1,x_2,x_3)=(\sin x_1)^{\cos x_2}+2^{x_3^2-x_1x_2}$	$f(z_1,z_2,z_3)=(z_3\cup z_1)\cap(z_1\cup\bar{z}_2)\cap(\bar{z}_2\cup\bar{z}_3)$
2	$y(x_1,x_2,x_3)=\arccos\frac{x_1x_3^2}{\sqrt{x_1+x_2}}$	$f(z_1,z_2,z_3)=(\bar{z}_1\cup z_3)\cap(z_2\cup\bar{z}_3)\cap(\bar{z}_1\cup z_2)$
3	$y(x_1,x_2,x_3)=\arcsin\frac{x_3}{\sqrt{x_1^2+x_2^2+x_3^2}}$	$f(z_1,z_2,z_3)=\overline{\bar{z}_1\cap z_3}\cup z_2\cap\bar{z}_3\cap(\bar{z}_1\cup z_2)$
4	$y(x_1,x_2,x_3)=\cos\frac{x_3x_1}{x_2}+5^{x_2^2+3x_1-x_3}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap(z_2\cup\bar{z}_3)\cup\overline{(\bar{z}_1\cup z_2)}$
5	$y(x_1,x_2,x_3)=\sin\frac{x_2x_1^2}{x_2+x_3}+3^{2x_2+x_3-x_1}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_2)}\cap(z_2\cup\bar{z}_3)\cup\bar{z}_1\cap\bar{z}_3$
6	$y(x_1,x_2,x_3)=\arccos\frac{x_3}{\sqrt{x_1^2+x_2^2}}$	$f(z_1,z_2,z_3)=\overline{\overline{(\bar{z}_1\cup z_2)}\cap\overline{(\bar{z}_1\cup\bar{z}_3)}}\cup\bar{z}_1\cap\bar{z}_3$
7	$y(x_1,x_2,x_3)=(tgx_1)^{\sin x_2}+5^{x_3^2}$	$f(z_1,z_2,z_3)=z_3\cap z_1\cup z_1\cap\bar{z}_3\cup\overline{(\bar{z}_2\cup z_3)}$
8	$y(x_1,x_2,x_3)=arctg\frac{x_1+x_2}{1+x_1x_2}+x_3$	$f(z_1,z_2,z_3)=(\bar{z}_3\cup z_1)\cap\overline{(z_1\cup\bar{z}_2)}\cup\bar{z}_2\cap z_3$
9	$y(x_1,x_2,x_3)=\arcsin\frac{2x_2}{\sqrt{x_1^2+x_3^2}}$	$f(z_1,z_2,z_3)=(\bar{z}_2\cup z_3)\cap\overline{(z_1\cup\bar{z}_3)}\cup\overline{(\bar{z}_1\cup z_2)}$
10	$y(x_1,x_2,x_3)=\sin\frac{x_1x_3^2}{x_1+x_2}+4^{x_1+2x_2-x_3}$	$f(z_1,z_2,z_3)=z_2\cap\bar{z}_3\cup\overline{(\bar{z}_1\cup z_3)\cap(\bar{z}_1\cup\bar{z}_2)}$
11	$y(x_1,x_2,x_3)=\lg\sqrt[3]{\frac{x_1^2+x_2+1}{x_1x_3}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_2\cup z_1)}\cap(z_2\cup\bar{z}_3)\cap(\bar{z}_1\cup\bar{z}_3)$
12	$y(x_1,x_2,x_3)=tg\frac{x_1+x_2}{x_3^2}+3^{x_1^3-2x_1x_2+x_3}$	$f(z_1,z_2,z_3)=(\bar{z}_1\cup z_2)\cap\overline{(z_1\cup\bar{z}_3)\cap(\bar{z}_2\cup z_3)}$

13	$y(x_1,x_2,x_3)=\cos\frac{x_3}{(x_1+x_2)^2}-x_3^{x_1-x_2}$	$f(z_1,z_2,z_3)=\overline{\overline{\bar{z}_1\cap z_2}\cup z_3}\cap\bar{z}_2\cup(\bar{z}_1\cup z_3)$
14	$y(x_1,x_2,x_3)=\sqrt[8]{9+x_1^2+x_2}-\frac{x_2}{\sqrt{4x_1^2+x_3^2}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap\overline{(z_1}\cup z_3)\cup\overline{(\bar{z}_2\cup z_3)}$
15	$y(x_1,x_2,x_3)=\frac{\sin(x_1x_2)}{x_1^2+x_2^2}-3x_3^{x_1+x_2}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_3)}\cap(z_2\cup\bar{z}_3)\cup\overline{\bar{z}_1\cap\bar{z}_2}$
16	$y(x_1,x_2,x_3)=x_1^{\sin x_2}\cdot(x_1+x_3^2)+x_1\lg(x_3^2+3x_2)$	$f(z_1,z_2,z_3)=\overline{\overline{(\bar{z}_2\cup z_3)}\cap(\bar{z}_1\cup\bar{z}_2)}\cup\bar{z}_1\cap z_3$
17	$y(x_1,x_2,x_3)=\sqrt{\frac{x_1^2+x_2}{x_3^2}}+\lg x_1$	$f(z_1,z_2,z_3)=z_2\cap z_3\cup\bar{z}_1\cap\bar{z}_3\cup\overline{(z_1\cup\bar{z}_2)}$
18	$y(x_1,x_2,x_3)=\frac{\lg(x_1+x_2)}{\sqrt{x_1^2+2x_2+x_3}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_2\cup z_1)}\cap\overline{(z_2\cup\bar{z}_3)}\cup\bar{z}_1\cap z_3$
19	$y(x_1,x_2,x_3)=x_1^{x_2}\cdot\lg(x_1+x_2+x_3^2)$	$f(z_1,z_2,z_3)=(z_1\cup\bar{z}_3)\cap\overline{(z_1\cup\bar{z}_2)}\cup\overline{(z_2\cup\bar{z}_3)}$
20	$y(x_1,x_2,x_3)=\frac{\lg(x_1^2+x_2^2-8)}{\sqrt{x_2^2+x_1^2-4x_3}}$	$f(z_1,z_2,z_3)=\bar{z}_1\cap z_2\cup\overline{(z_2\cup\bar{z}_3)\cap(z_1\cup\bar{z}_3)}$

Контрольні питання

1. Які етапи створення додатків у середовищі ISaGRAF?
2. Яке призначення має мова програмування FBD і коли вона застосовується?
3. Що таке FBD-діаграма?
4. Яка структура функціонального блоку?
5. Що таке змінна в програмі FBD і які види змінних бувають?
6. Які знаєте функціональні блоки, що реалізують арифметичні операції?
7. Які знаєте функціональні блоки, що реалізують стандартні математичні функції?
8. Які ви знаєте функціональні блоки, що реалізують логічні операції?
9. Яким чином у програмі на FBD вводиться "логічне заперечення" булевої змінної?.
10. Яке призначення має оператор RETURN в програмі FBD?
11. Коли використовуються стрибки і мітки в програмі FBD?

ЛАБОРАТОРНА РОБОТА № 2

Тема: ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ ISaGRAF НА МОВІ LD

Мета роботи: знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування LD.

Короткі теоретичні відомості

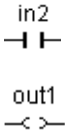
МОВА LD

Мова релейних діаграм (Ladder Diagram, LD) – це графічне представлення логічних рівнянь, комбінуючі **контакти** (входи) і **вітки** (виходи). Мова LD дозволяє описувати роботу з **булевими** даними, розміщаючи **графічні символи** в схему програми. Графічні символи LD організовані всередині схеми також, як електрична схема. Праворуч і ліворуч LD діаграма повинна єднатися з вертикальними силовими шинами. Основні компоненти LD діаграми:

 – ліва і права вертикальні силові шини;

 – горизонтальна і вертикальна лінії зв'язку;

 – множинні лінії з'єднання;



– контакт и виток, зв’язані зі змінними.

Для представлення контактів використовуються символи:

• Прямий контакт.

• Інвертований контакт.

• Контакт з визначенням переднього P і заднього N фронтів.

Для представлення витків використовуються символи:

• Прямий виток.

• Інвертований виток.

• SET виток S.

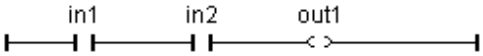
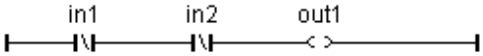
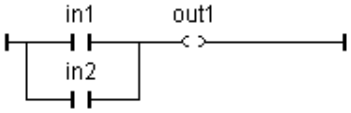
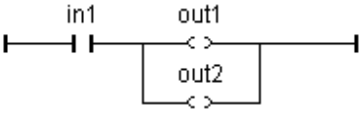
• RESET виток R.

• Виток з визначенням фронтів P, N.

Ім'я змінної пишеться над цими графічними символами.

Приклади використання компонентів LD діаграми приведені в таблиці 1.

Таблиця 1 – Приклади використання компонентів LD діаграми

Математичний запис	Програмна реалізація
$out1 = in1 \wedge in2$	
$out1 = \overline{in1} \wedge \overline{in2}$	
$out1 = in1 \vee in2$	
out1=in1 out2=in1	

В мові LD може бути використаний оператор RETURN, а також мітки і безумовні переходи.

Мітка RETURN може бути використана як вихід, щоб представити умовне завершення програми. Ніяких символів до правого кінця RETURN підключати не можна.

У редакторі LD можна підключати функціональні блоки до логічних ліній. Так як блоки не завжди мають логічні входи і (або) логічні виходи, введення блоків до LD діаграми приводить до додавання декількох нових параметрів EN, ENO в інтерфейсі блока (рисунок 1).

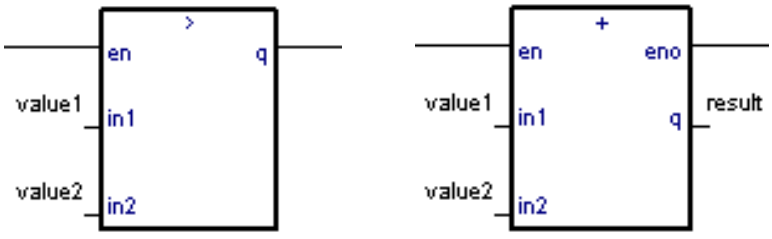


Рисунок 1 – Функціональні блоки EN та ENO

Так як перший вихід завжди повинен бути підключений до шини, на першу позицію автоматично вводиться інший вихід, названий "ENO". Вихід ENO завжди має теж значення, що і перший вхід блоку.

Так як перший вхід завжди повинен бути підключений до шини, на першу позицію автоматично вводиться інший вхід, названий "EN". Блок виконується тільки тоді, коли вхід EN дорівнює TRUE.

Елементи редактора мови LD

F2: Contact on the left: вставка контакту ліворуч від виділеного.

F3: Contact on the right: вставка контакту праворуч від виділеного.

F4: Parallel contact: вставка паралельного контакту.

F5: Coil: вставка вітка.

F6: Block on the left: вставка блоку ліворуч від виділеного.

F7: Block on the right: вставка блоку праворуч від виділеного.

- **Enter Symbol**: для введення функціональних блоків, з'являється вікно (рисунок 2).

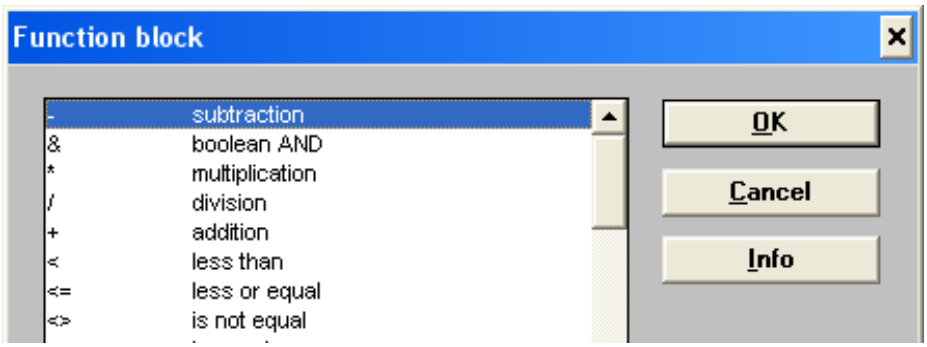


Рисунок 2 – Вікно вибору функціональних блоків

Coil/contact type: інвертування контакту і витка.

Cell width: стиснення кола.

Приклад виконання роботи

Постановка задачі. Розробити та дослідити додаток на мові LD для віртуального контролера, який реалізує обчислення наступних арифметичних і логічних виразів:

$$y(x_1, x_2, x_3) = \sin\frac{x_3 x_1^{\,2}}{x_3 + x_2},$$

$$f(z_1, z_2, z_3) = \overline{\bar{z}_1 \cap z_2} \cup z_3 \cap (\bar{z}_1 \cup \bar{z}_3),$$

де x_1, x_2, x_3 – вхідні дійсні змінні; y – вихідна дійсна змінна; z_1, z_2, z_3 – вхідні булеві змінні; f – вихідна булева змінна.

Рішення задачі

1. Створити новий проєкт з ім'ям "project2" в системі ISaGRAF.

2. Створити нову програму "program2". При виборі мови вказати мову LD (рисунок 3).

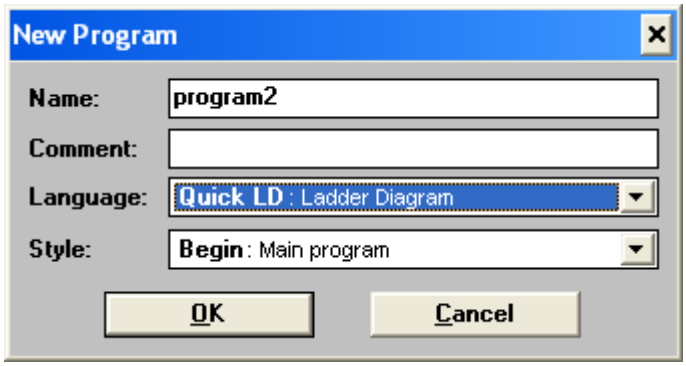


Рисунок 3 – Вибір мови LD

3. Оголосити використовувані змінні за допомогою команди "Dictionary (Словник)" меню "File" або кнопки  (рисунок 4).

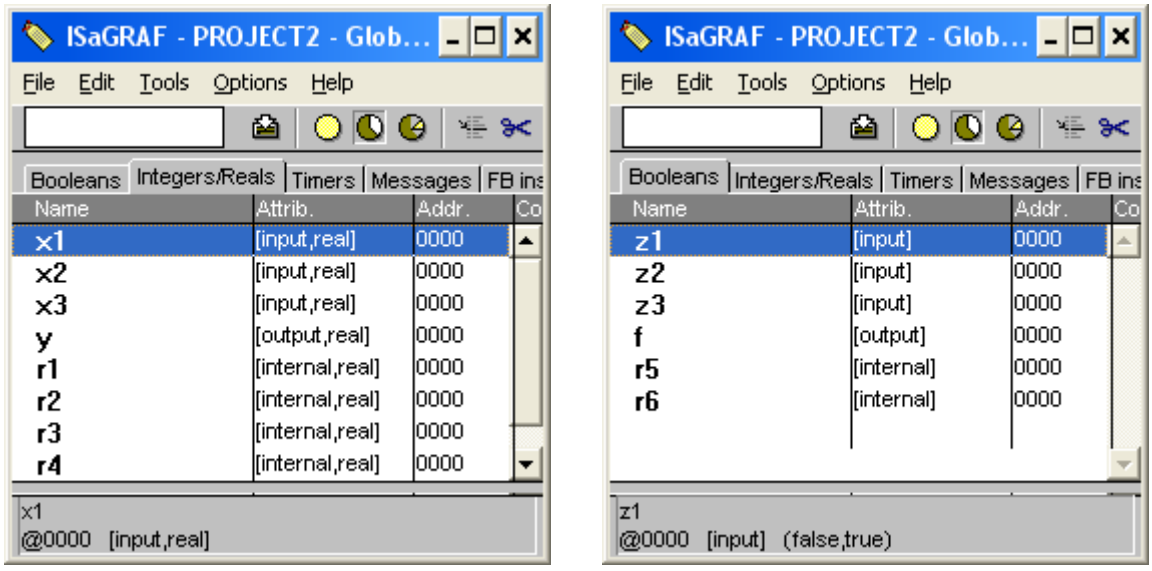


Рисунок 4 – Визначення змінних

4. Відредагувати програму відповідно до умови задачі.

У вікні редактора натиснути **F6: Block on the left** (рисунок 5).

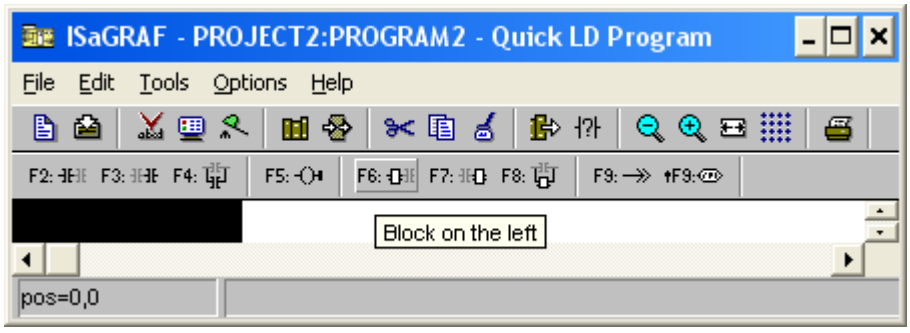


Рисунок 5 – Виконання команди вставки блоку

З'явиться ланцюг (рисунок 6).

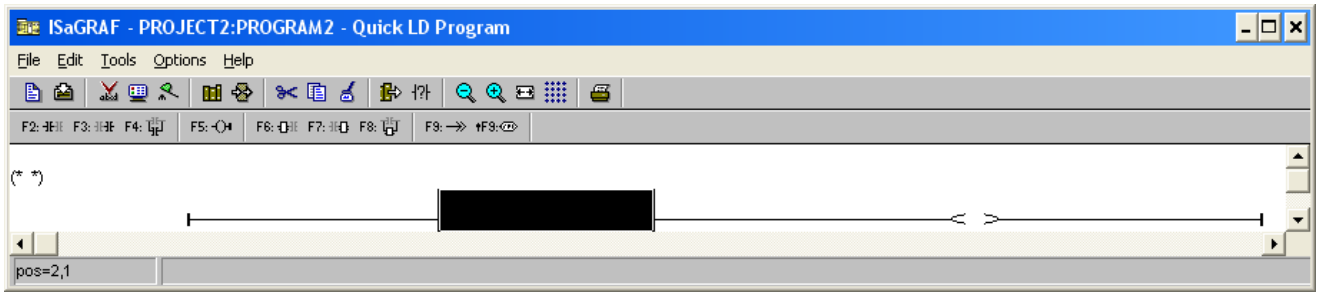


Рисунок 6 – Створення першого ланцюга програми

Клацнувши 2ЛКМ по (* *), можна ввести текст коментарю у вікні, що з'явилося (рисунок 7).

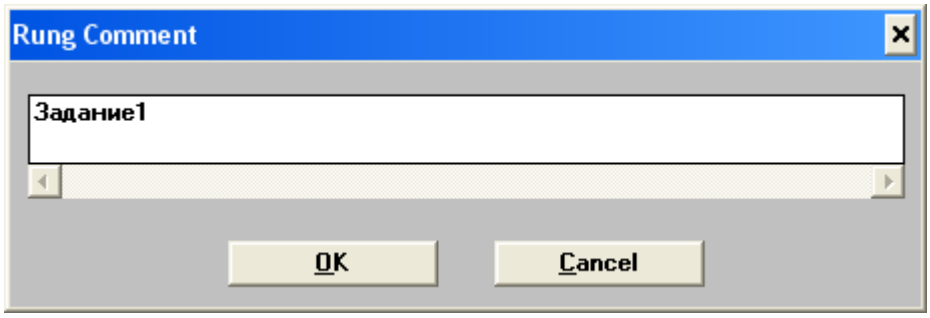


Рисунок 7 – Створення коментарю

Вставити ще один блок, натиснувши **F7: Block on the right** (рисунок 8).

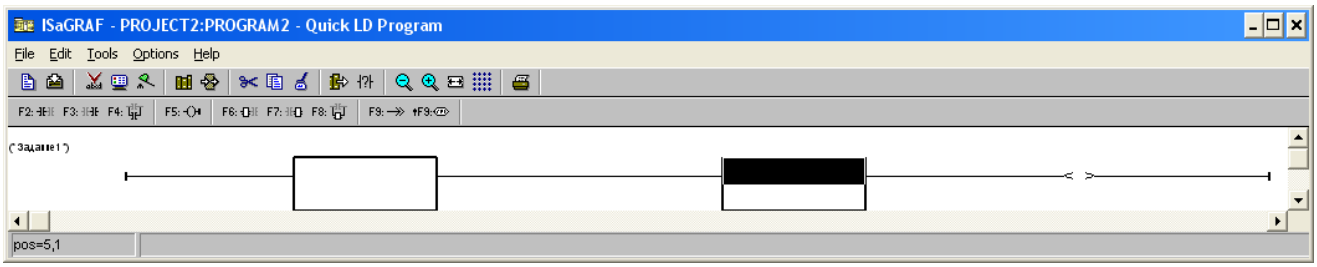


Рисунок 8 – Вставка другого блоку

Виділяємо перший блок і, натиснувши кнопку **Enter Symbol**, у вікні вибираємо потрібну операцію (рисунок 9).

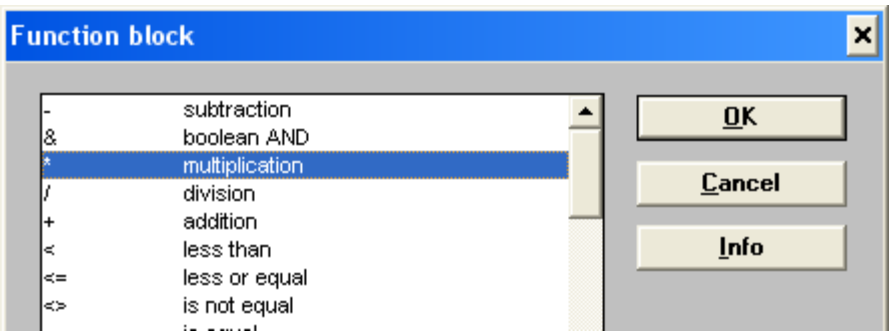


Рисунок 9 – Вибір операції

Аналогічно - для другого блоку. Щоб стиснути ланцюг, натисніть кнопку **Cell width** (рисунок 10).

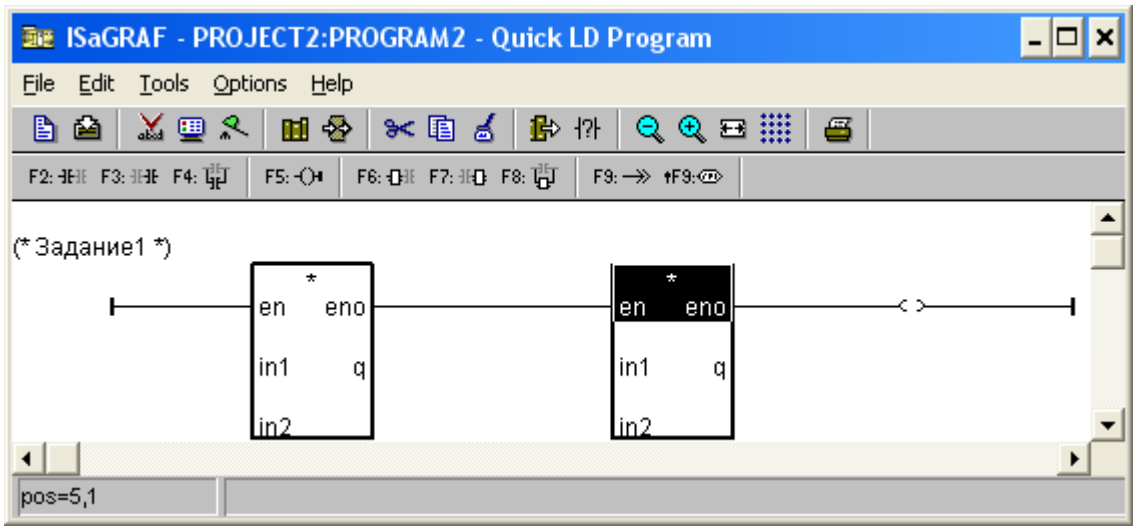


Рисунок 10 – Стиснення ланцюга

Виділити вхід **in1** першого блоку, натиснути кнопку **Enter Symbol** і вибрати потрібну змінну. Після прив'язки усіх входів і виходів отримаємо перший ланцюг, який обчислює чисельник аргументу функції (рисунок 11).

Для створення другого ланцюга ставимо курсор нижче першого ланцюга і аналогічним чином вставляємо блоки для виконання інших операцій. В результаті отримаємо програму першого завдання на мові LD (рисунок 12).

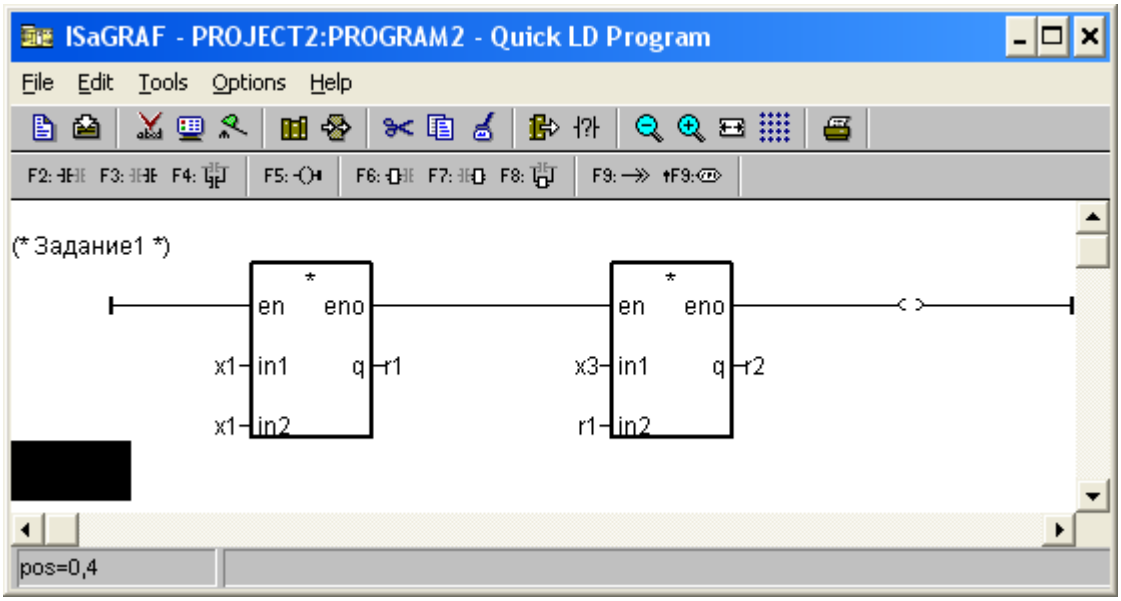


Рисунок 11 – Прив'язка входів і виходів першого ланцюга

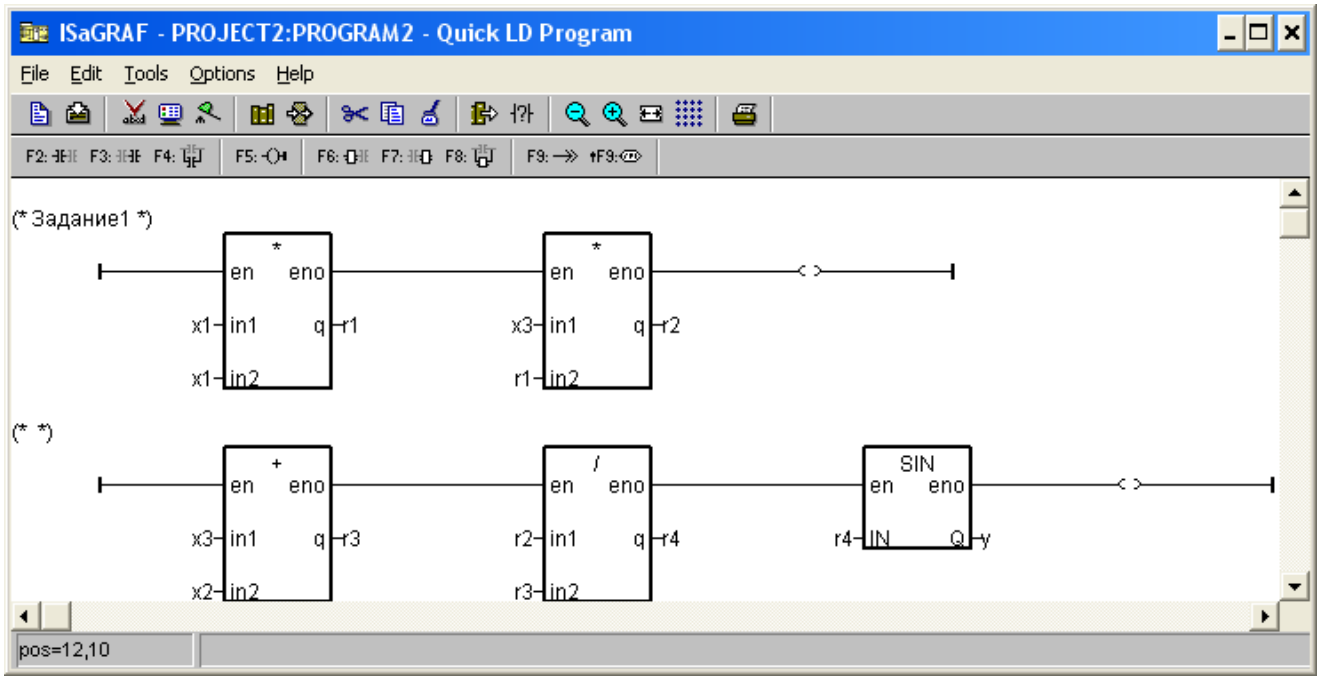


Рисунок 12 – Програма першого завдання

Для складання програми завдання 2 треба поставити курсор нижче другого ланцюга і натиснути кнопку **F2 : Contact on the left**. Аналогічно можна вставити коментар (рисунок 13).

Щоб зробити контакт інверсним, виділіть його і натисніть кнопку **Coil /contact type** (рисунок 14).

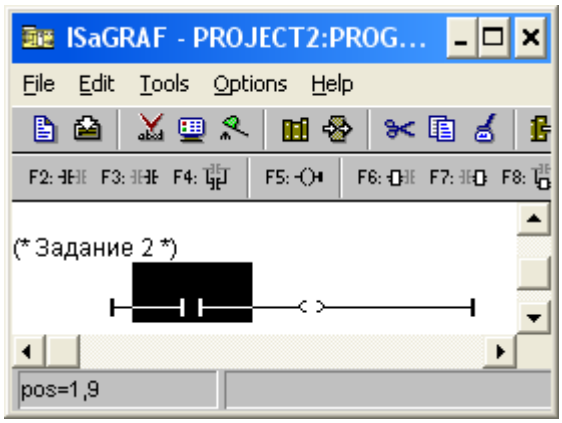


Рисунок 13 – Створення програми другого завдання

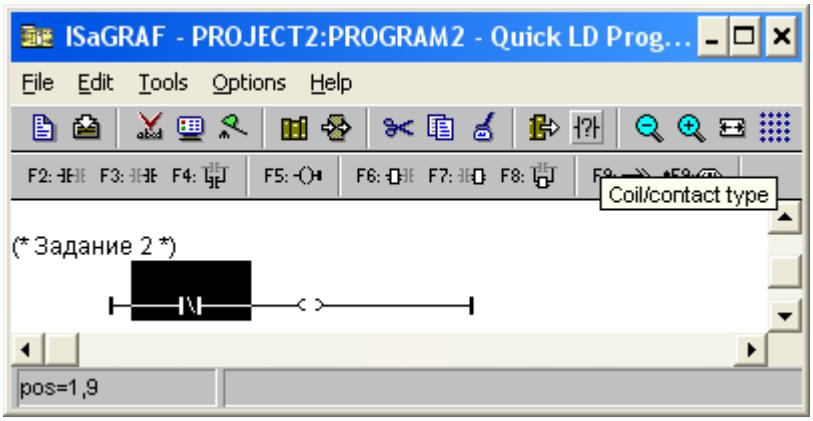


Рисунок 14 – Інверсія контакту

При виділеному першому контакті вставляємо другий контакт: **F3: Contact on the right**. Виділяємо виток і також його інвертуємо (за умовою) (рисунок 15).

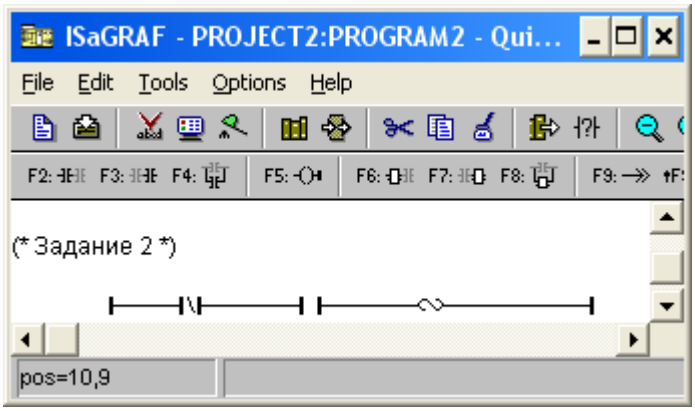


Рисунок 15 – Перший ланцюг другого завдання

Виділяємо по черзі контакти і виток і вибираємо для них потрібні змінні (рисунок 16).

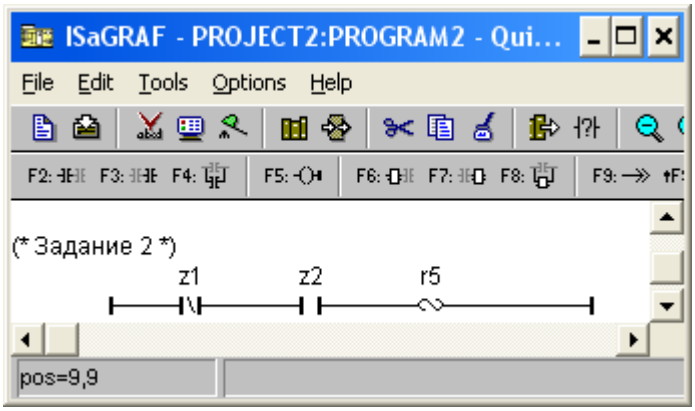


Рисунок 16 – Прив’язка змінних до першого ланцюга завдання 2

Аналогічним чином будуємо інші ланцюги завдання 2 (рисунок 17).

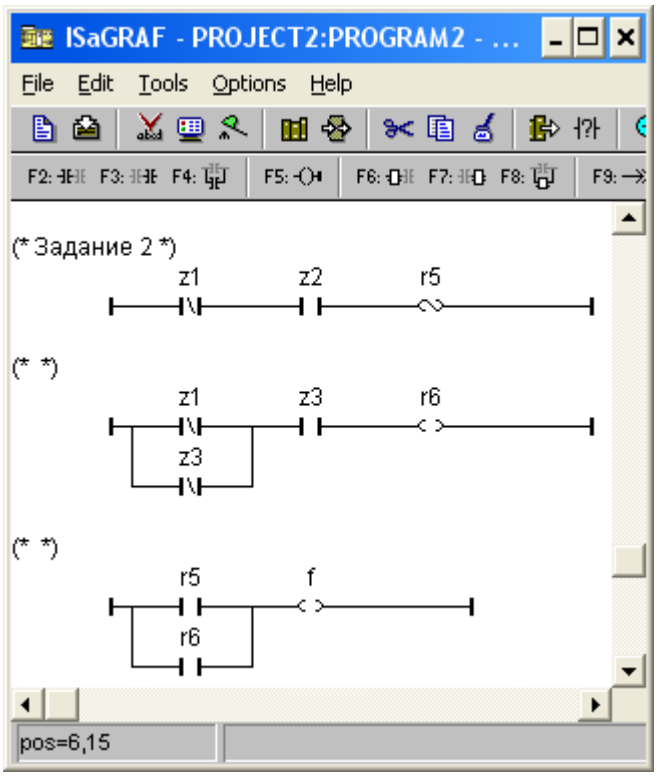


Рисунок 17 – Програма другого завдання

5. Настроїти конфігурацію введення/виведення і здійснити прив'язку вхідних і вихідних змінних проєкту.

6. Створити код додатка.

7. Провести налагодження додатка в режимі симуляції.

Виконання п.1 - 3, 5 - 7 детально розглянуто в лабораторній роботі № 1.

8. Виконати тестування і перевірку отриманих результатів для свого варіанту, як в лабораторній роботі № 1.

Тест № 1

Результати тестування приведені на рисунках 18 – 19.

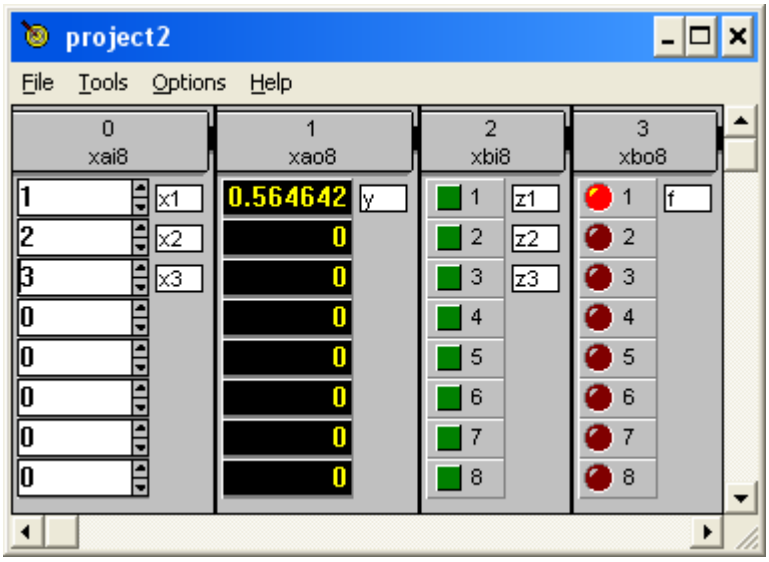


Рисунок 18 – Вікно симулятора з результатом тесту №1

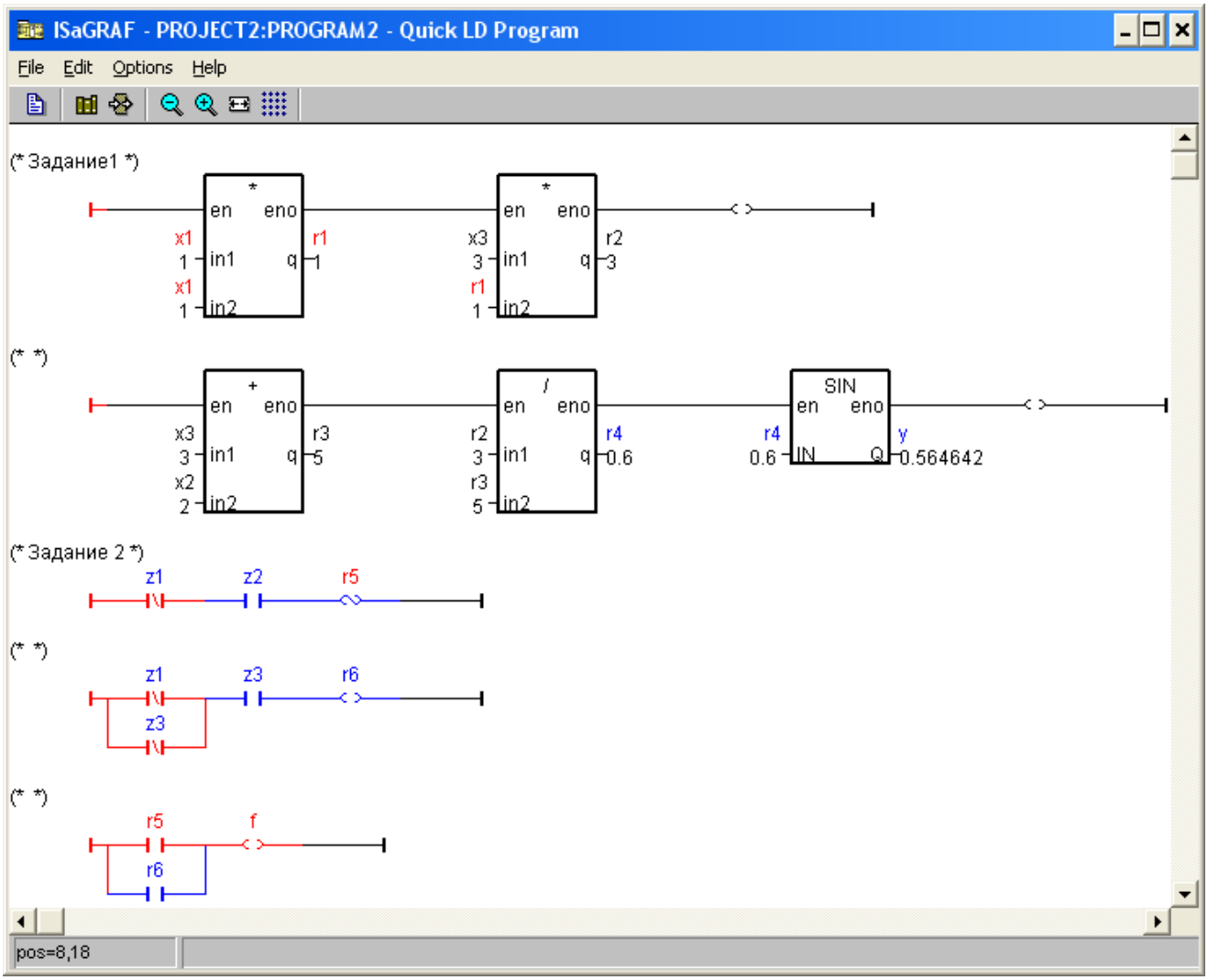


Рисунок 19 – Програма в режимі online

Тест № 2

Результати тестування приведені на рисунках 20 – 21.

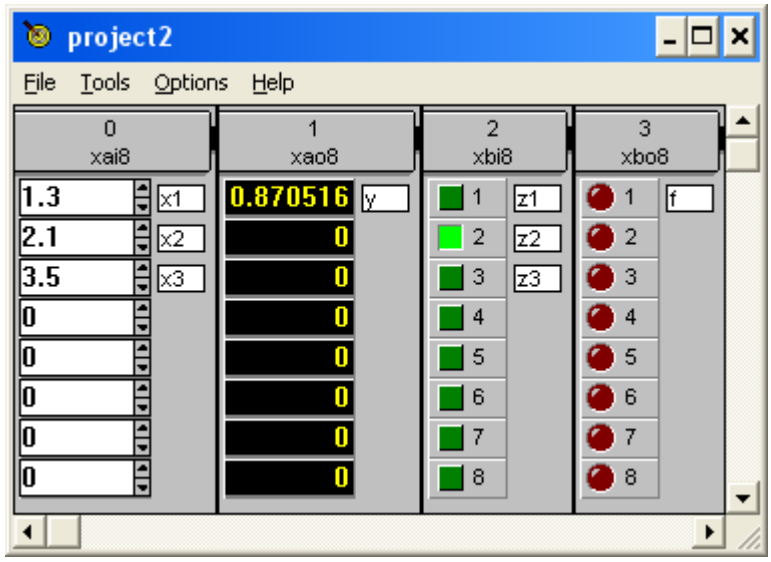


Рисунок 20 – Вікно симулятора з результатом тесту №2

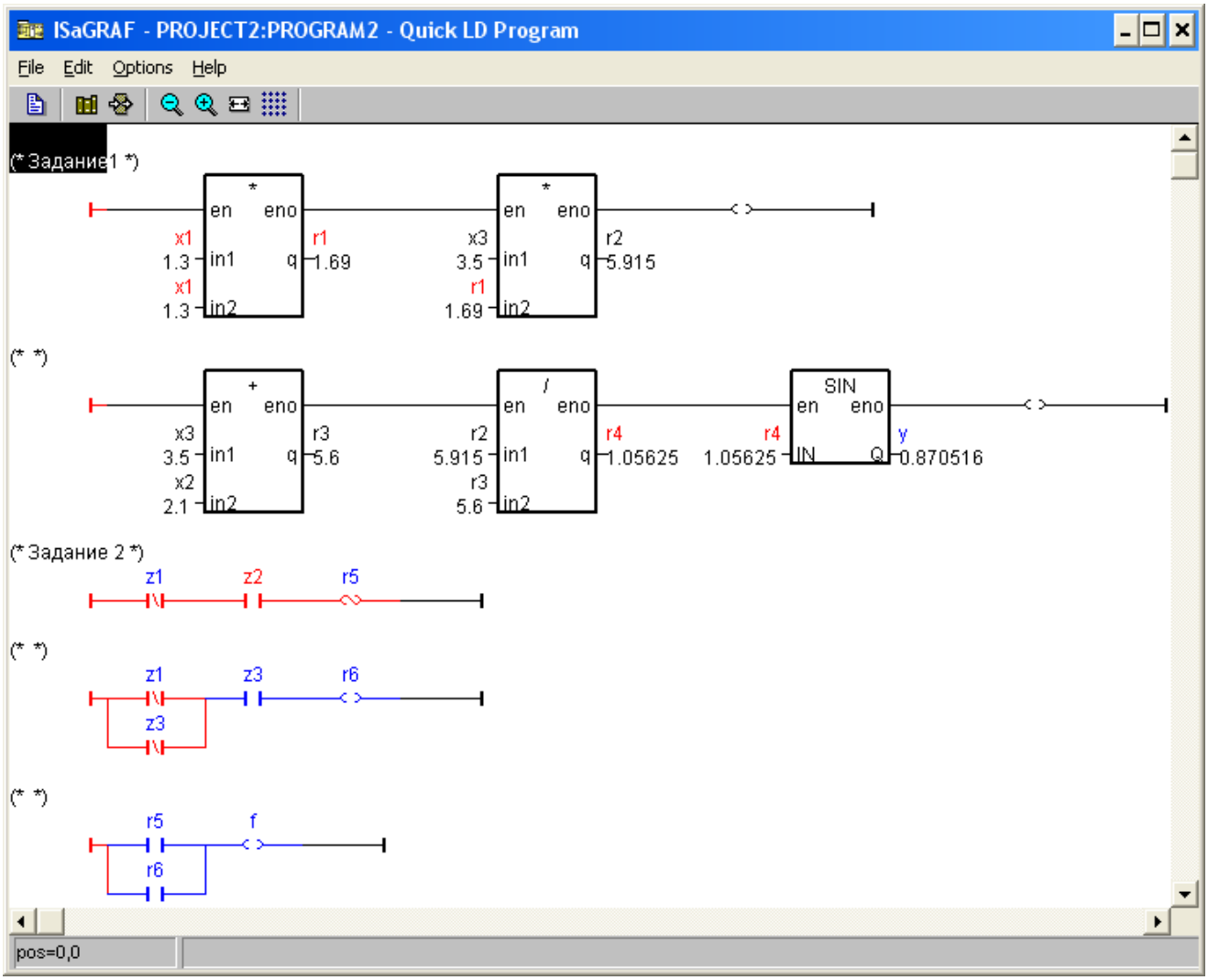


Рисунок 21 – Результат виконання програми

Тест № 3

Результати тестування приведені на рисунках 22 – 23.

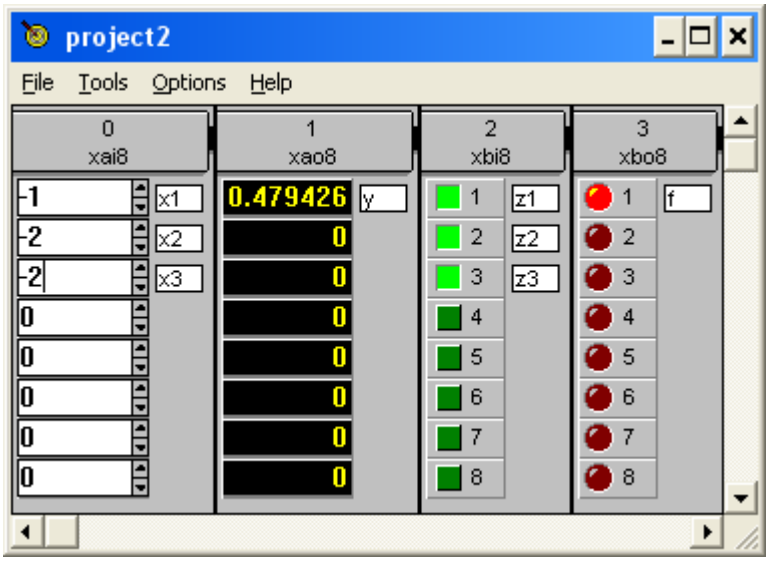


Рисунок 22 – Вікно симулятора з результатом тесту №3

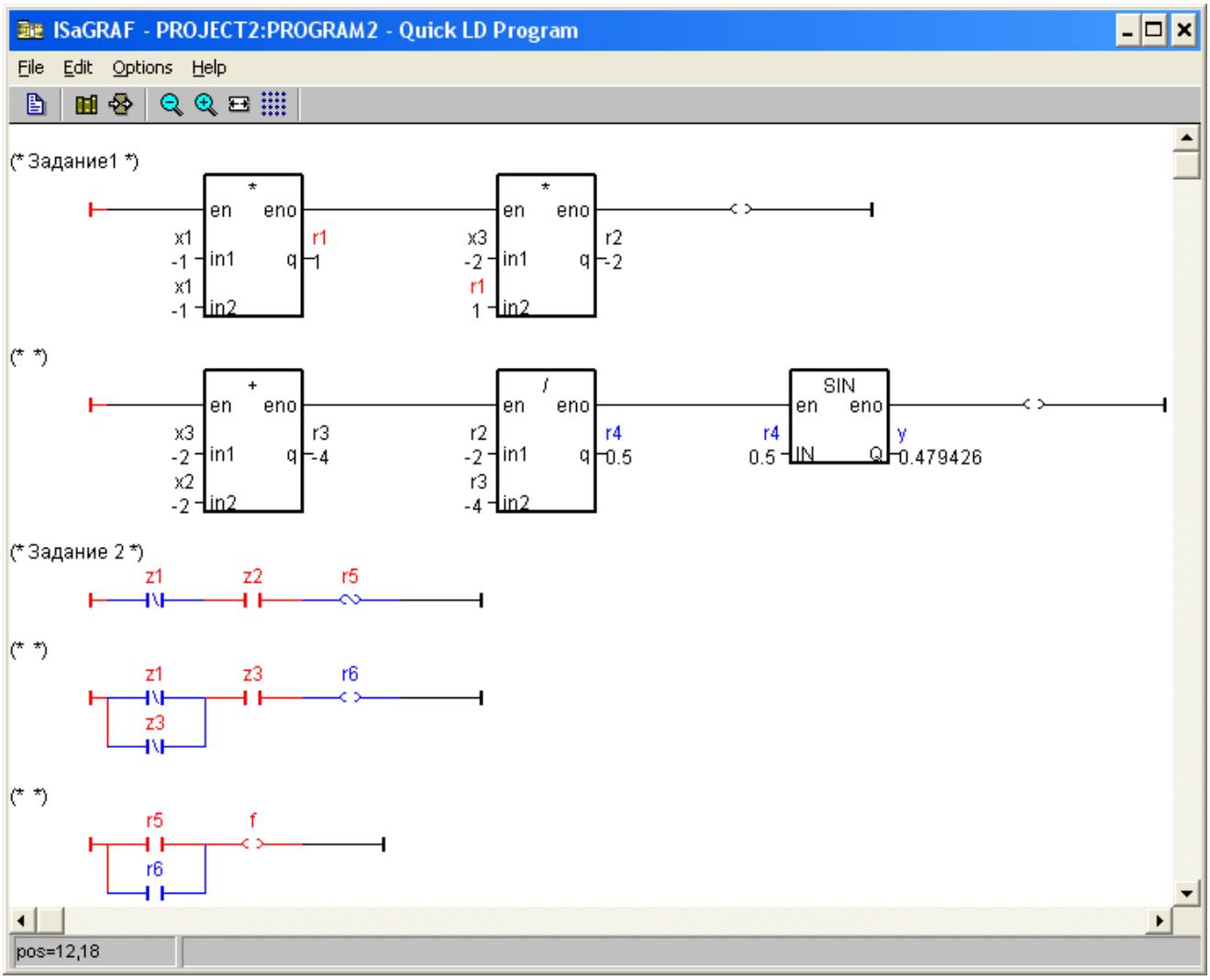


Рисунок 23 – Результат виконання програми

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 2.

Таблиця 2 – Варіанти завдань

№	Завдання 1	Завдання 2
1	$y(x_1,x_2,x_3)=\arccos\frac{x_1{x_3}^2}{\sqrt{x_1+x_2}}$	$f(z_1,z_2,z_3)=z_2\cap\bar{z}_3\cup\overline{(\bar{z}_1\cup z_3)\cap(\bar{z}_1\cup\bar{z}_2)}$
2	$y(x_1,x_2,x_3)=\arcsin\frac{x_3}{\sqrt{{x_1}^2+{x_2}^2+{x_3}^2}}$	$f(z_1,z_2,z_3)=(z_3\cup z_1)\cap(z_1\cup\bar{z}_2)\cap(\bar{z}_2\cup\bar{z}_3)$
3	$y(x_1,x_2,x_3)=\cos\frac{x_3x_1}{x_2}+5^{{x_2}^2+3x_1-x_3}$	$f(z_1,z_2,z_3)=(\bar{z}_1\cup z_3)\cap(z_2\cup\bar{z}_3)\cap(\bar{z}_1\cup z_2)$
4	$y(x_1,x_2,x_3)=\sin\frac{x_2{x_1}^2}{x_2+x_3}+3^{2x_2+x_3-x_1}$	$f(z_1,z_2,z_3)=\overline{\bar{z}_1\cap z_3}\cup z_2\cap\bar{z}_3\cap(\bar{z}_1\cup z_2)$
5	$y(x_1,x_2,x_3)=\arccos\frac{x_3}{\sqrt{{x_1}^2+{x_2}^2}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap(z_2\cup\bar{z}_3)\cup\overline{(\bar{z}_1\cup z_2)}$
6	$y(x_1,x_2,x_3)=(tgx_1)^{\sin x_2}+5^{{x_3}^2}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_2)}\cap(z_2\cup\bar{z}_3)\cup\bar{z}_1\cap\bar{z}_3$
7	$y(x_1,x_2,x_3)=arctg\frac{x_1+x_2}{1+x_1x_2}+x_3$	$f(z_1,z_2,z_3)=\overline{\overline{(\bar{z}_1\cup z_2)}\cap(\bar{z}_1\cup\bar{z}_3)}\cup\bar{z}_1\cap\bar{z}_3$
8	$y(x_1,x_2,x_3)=\arcsin\frac{2x_2}{\sqrt{{x_1}^2+{x_3}^2}}$	$f(z_1,z_2,z_3)=z_3\cap z_1\cup z_1\cap\bar{z}_3\cup\overline{(\bar{z}_2\cup z_3)}$
9	$y(x_1,x_2,x_3)=\sin\frac{x_1{x_3}^2}{x_1+x_2}+4^{x_1+2x_2-x_3}$	$f(z_1,z_2,z_3)=(\bar{z}_3\cup z_1)\cap\overline{(z_1\cup\bar{z}_2)}\cup\bar{z}_2\cap z_3$
10	$y(x_1,x_2,x_3)=\lg\sqrt{\frac{{x_1}^2+x_2+1}{x_1x_3}}$	$f(z_1,z_2,z_3)=(\bar{z}_2\cup z_3)\cap\overline{(z_1\cup\bar{z}_3)}\cup\overline{(\bar{z}_1\cup z_2)}$
11	$y(x_1,x_2,x_3)=tg\frac{x_1+x_2}{{x_3}^2}+3^{{x_1}^3-2x_1x_2+x_3}$	$f(z_1,z_2,z_3)=(\bar{z}_1\cup z_2)\cap\overline{(z_1\cup\bar{z}_3)\cap(\bar{z}_2\cup z_3)}$
12	$y(x_1,x_2,x_3)=\cos\frac{x_3}{(x_1+x_2)^2}-{x_3}^{x_1-x_2}$	$f(z_1,z_2,z_3)=\overline{\overline{\bar{z}_1\cap z_2}\cup z_3}\cap\bar{z}_2\cup(\bar{z}_1\cup z_3)$
13	$y(x_1,x_2,x_3)=\sqrt[8]{9+{x_1}^2+x_2}-\frac{x_2}{\sqrt{4{x_1}^2+{x_3}^2}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap\left(\overline{z_1}\cup z_3\right)\cup\overline{(\bar{z}_2\cup z_3)}$
14	$y(x_1,x_2,x_3)=\frac{\sin(x_1x_2)}{{x_1}^2+{x_2}^2}-3{x_3}^{x_1+x_2}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_3)}\cap(z_2\cup\bar{z}_3)\cup\overline{\bar{z}_1\cap\bar{z}_2}$
15	$y(x_1,x_2,x_3)={x_1}^{\sin x_2}\cdot(x_1+{x_3}^2)+x_1\lg({x_3}^2+3x_2)$	$f(z_1,z_2,z_3)=\overline{\overline{(\bar{z}_2\cup z_3)}\cap(\bar{z}_1\cup\bar{z}_2)}\cup\bar{z}_1\cap z_3$

16	$y(x_1, x_2, x_3) = \sqrt{\frac{x_1^{\,2} + x_2}{x_3^{\,2}}} + \lg x_1$	$f(z_1, z_2, z_3) = z_2 \cap z_3 \cup \bar{z}_1 \cap \bar{z}_3 \cup \overline{(z_1 \cup \bar{z}_2)}$
17	$y(x_1, x_2, x_3) = \frac{\lg(x_1 + x_2)}{\sqrt{x_1^{\,2} + 2x_2 + x_3}}$	$f(z_1, z_2, z_3) = \overline{(\bar{z}_2 \cup z_1)} \cap \overline{(z_2 \cup \bar{z}_3)} \cup \bar{z}_1 \cap z_3$
18	$y(x_1, x_2, x_3) = x_1^{\,x_2} \cdot \lg(x_1 + x_2 + x_3^{\,2})$	$f(z_1, z_2, z_3) = (z_1 \cup \bar{z}_3) \cap \overline{(z_1 \cup \bar{z}_2)} \cup \overline{(z_2 \cup \bar{z}_3)}$
19	$y(x_1, x_2, x_3) = \frac{\lg(x_1^{\,2} + x_2^{\,2} - 8)}{\sqrt{x_2^{\,2} + x_1^{\,2} - 4x_3}}$	$f(z_1, z_2, z_3) = \bar{z}_1 \cap z_2 \cup \overline{(z_2 \cup \bar{z}_3) \cap (z_1 \cup \bar{z}_3)}$
20	$y(x_1, x_2, x_3) = \frac{2x_1 - 3x_2}{\lg(x_1 + 2)^2} - \sqrt{x_1 + x_2^{\,2} + 4x_3}$	$f(z_1, z_2, z_3) = \overline{(z_3 \cup z_2)} \cap (z_2 \cup \bar{z}_1) \cap (\bar{z}_1 \cup \bar{z}_3)$

Контрольні питання

1. Яке основне призначення мови програмування LD і в яких випадках вона застосовується?

2. Шини живлення та сполучні лінії LD діаграми.

3. Які основні елементи мови LD?

4. Які види контактів є в мові LD?

5. У яких випадках при створенні програми на LD використовується послідовне з'єднання контактів, а в яких паралельне?

6. Які види витків є в мові LD?

7. Множинне з'єднання LD діаграми.

8. Оператор RETURN мові LD.

9. Як підключаються функціональні блоки до LD діаграми?

10. Елементи редактора мови LD.

ЛАБОРАТОРНА РОБОТА № 3

Тема: ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ ISaGRAF НА МОВІ ST

Мета роботи: знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування ST.

Короткі теоретичні відомості

МОВА ST

Мова **структурованого тексту** (Structured text, ST) – це структурна текстова мова високого рівня. Ця мова використовується для створення складних процедур, які не можуть бути легко виражені за допомогою графічних мов. За замовчуванням ST є мовою для опису дій всередині кроків і умов мови **SFC.**

ST-програма – це список ST-операторів. Кожен оператор закінчується крапкою з комою (;).

Основні оператори мови ST: оператор **присвоєння** (variable := expression); виклик підпрограми і функції; виклик **функціонального блока**; оператори **вибору** (IF, THEN, ELSE, CASE); **ітеративні** оператори (FOR, WHILE, REPEAT); **керуючі оператори** (RETURN, EXIT); спеціальні оператори для зв'язку з такими мовами, як **SFC.**

Виклик функцій в ST здійснюється у відповідності з наступним синтаксисом:

<rezult>:=<function>(<par1>, <par2>,…);

де **<rezult>** – ім'я змінної, якій присвоюється результат виконання функції; **<function>** – ім'я викликаємої функції; **<par1>, <par2>,…** – список операндів. Наприклад:

```
y := sin(x1);
z := real(100*x2);
```

Використання в ST функціональних блоків проводиться у відповідності з наступним синтаксисом:

<blockname> (<p1>, <p2> ...);

<result_1> := <blockname>.<ret_param1>;

... **<result_N> := <blockname>.<ret_paramN>;**

де **<blockname>** – ім'я екземпляра викликаємого функціонального блоку (перед використанням екземпляр функціонального блоку має бути визначений в словнику); **<result_1>,** … **<result_N>** – змінні, яким присвоюються результати виконання блоку; **<par1>, <par2>,…** – список операндів; **>; <ret_param1>,** …**,<ret_paramN>** – вихідні поля функціонального блоку. Наприклад:

blink1(run, t#2s);

y := blink1.q;

де **blink1** – екземпляр функціонального блоку **blink**; **q** – вихідне поле блоку **blink** (найменування вихідних полів слід дивитися в описі функціонального блоку).

Оператор **IF** призначений для організації розгалужень, при цьому виконуються 1 або 2 списку ST-операторів. Вибір здійснюється відповідно до значення булевского вираження. Синтаксис оператора IF має вид:

IF < boolean_expression > THEN

<statement > ;

<statement > ;

…

ELSIF <boolean_expression> THEN

<statement> ;

<statement> ;

…

ELSE

<statement> ;

<statement> ;

…

END_IF;

Оператори ELSE и ELSIF – додаткові. Якщо ELSE опущений і умова дорівнює FALSE, то ніяких інструкцій не виконується.

Оператор **CASE** виконує один або декілька списків ST-операторів, вибір здійснюється у відповідності з цілим виразом.

Синтаксис оператора **CASE** имає вид:

CASE <integer_expression> OF

<value>: <statements> ;

<value> , <value>: <statements> ;

…

ELSE

<statements> ;

END_CASE;

Оператор **WHILE** – оператор циклу з передумовою і має вигляд:

WHILE ... DO ... END_WHILE

Призначення: ітераційна структура для групи ST-операторів.
Умова "продовження" оцінюється ПЕРЕД будь-якою ітерацією.

Синтаксис: **WHILE <Boolean_expression>** DO
<statement> ;
<statement> ;
...
END_WHILE ;

Оператор **REPEAT** – оператор циклу з післямовою і має вигляд:

REPEAT ... UNTIL ... END_REPEAT

Призначення: ітераційна структура для групи ST-операторів. Умова "продовження" оцінюється ПІСЛЯ будь ітерації.

Синтаксис: REPEAT
<statement> ;
<statement> ;
...

UNTIL <Boolean_condition>

END_REPEAT ;

Оператор **FOR** використовується для організації циклів і має наступний вигляд:

FOR ... TO ... BY ... DO ... END_FOR

Призначення: виконує обмежену кількість ітерацій, використовуючи цілу індексний змінну.

Синтаксис: **FOR <index> := <mini> TO <maxi> BY <step> DO**

<statement> ;

<statement> ;

END_FOR;

Операнди: **index**: внутрішня ціла змінна, що збільшується в кожному циклі ітерації;

mini: початкове значення для індексу (перед першою итерацией);

maxi: максимально допустиме значення для індексу;

step: приріст індексу в кожному циклі ітерації.

Оператор [BY step] є необов'язковим. Якщо він не приведений, то крок збільшення дорівнює 1.

Приклад виконання роботи

Постановка задачі. Розробити та дослідити додаток на мові ST для віртуального контролера, який реалізує обчислення наступних арифметичних і логічних виразів:

$$y(x_1, x_2, x_3) = \sin\frac{x_3 x_1^{\,2}}{x_3 + x_2},$$

$$f(z_1, z_2, z_3) = \overline{\bar{z}_1 \cap z_2} \cup z_3 \cap (\bar{z}_1 \cup \bar{z}_3),$$

де x_1, x_2, x_3 – вхідні дійсні змінні; y – вихідна дійсна змінна; z_1, z_2, z_3 – вхідні булеві змінні; f – вихідна булева змінна.

Рішення задачі

1. Створити новий проєкт з ім’ям "project3" в системі ISaGRAF 3.4.

2. Створити нову програму "program3". При виборі мови указати мову ST (рисунок 1).

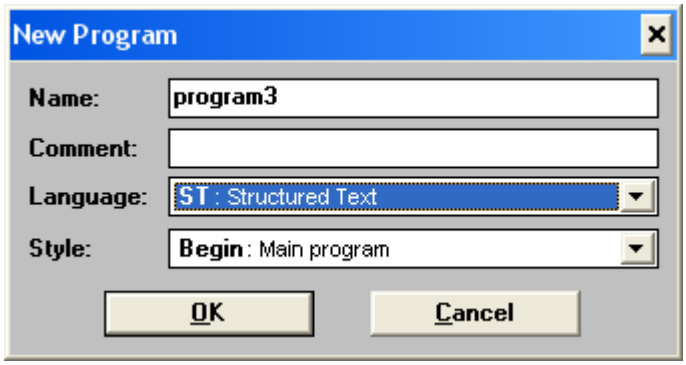


Рисунок 1 – Вибір мови ST

3. Об’явити необхідні змінні, згідно свого варіанту (таблиця 1).

4. Відредагувати програму відповідно до представленого нижче тексту (рисунок 2).

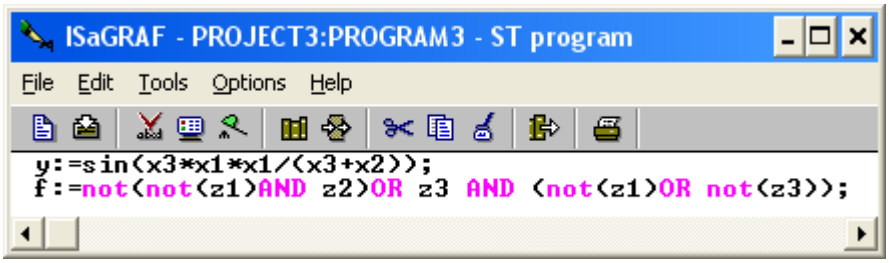


Рисунок 2 – Програма на мові ST

5. Настроїти конфігурацію введення/виведення і здійснити прив'язку вхідних і вихідних змінних проєкту.

6. Створити код додатку.

7. Провести налагодження додатку в режимі симуляції.

Виконання п.1 - 3, 5 - 7 детально розглянуто в лабораторній роботі № 1.

8. Виконати тестування і перевірку отриманих результатів для свого варіанту, як в лабораторній роботі № 1.

Тест № 1

Результат тестування представлений на рисунку 3.

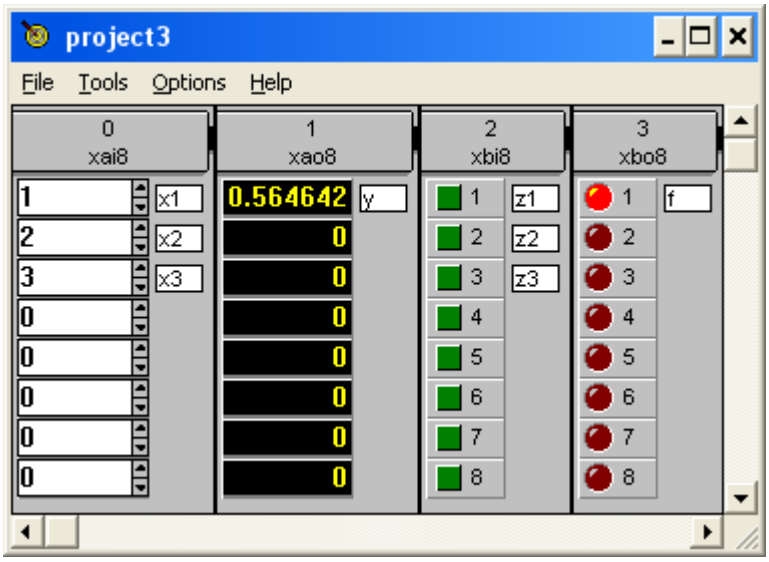


Рисунок 3 – Вікно симулятора з результатом тесту №1

Тест № 2

Результат тестування представлений на рисунку 4.

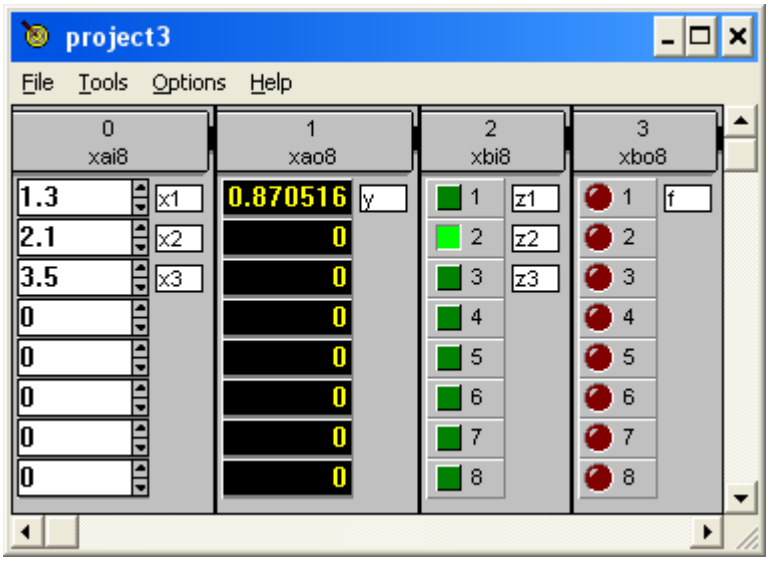


Рисунок 4 – Вікно симулятора з результатом тесту №2

Тест № 3

Результат тестування представлений на рисунку 5.

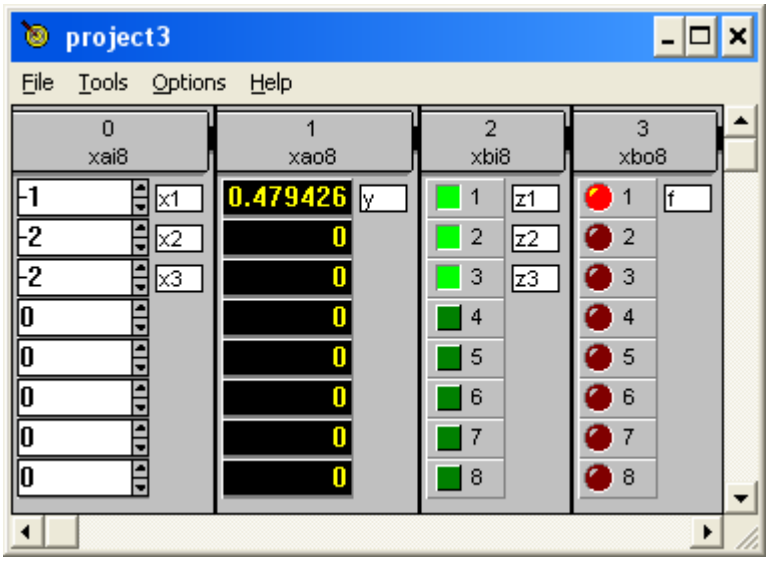


Рисунок 5 – Вікно симулятора з результатом тесту №3

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№	Завдання 1	Завдання 2
1	$y(x_1,x_2,x_3)=\arcsin\frac{x_3}{\sqrt{x_1^2+x_2^2+x_3^2}}$	$f(z_1,z_2,z_3)=(\bar{z}_2\cup z_3)\cap\overline{(z_1\cup\bar{z}_3)}\cup\overline{(\bar{z}_1\cup z_2)}$
2	$y(x_1,x_2,x_3)=\cos\frac{x_3x_1}{x_2}+5^{x_2^2+3x_1-x_3}$	$f(z_1,z_2,z_3)=z_2\cap\bar{z}_3\cup\overline{(\bar{z}_1\cup z_3)}\cap\overline{(\bar{z}_1\cup\bar{z}_2)}$
3	$y(x_1,x_2,x_3)=\sin\frac{x_2x_1^2}{x_2+x_3}+3^{2x_2+x_3-x_1}$	$f(z_1,z_2,z_3)=(z_3\cup z_1)\cap(z_1\cup\bar{z}_2)\cap(\bar{z}_2\cup\bar{z}_3)$
4	$y(x_1,x_2,x_3)=\arccos\frac{x_3}{\sqrt{x_1^2+x_2^2}}$	$f(z_1,z_2,z_3)=(\bar{z}_1\cup z_3)\cap(z_2\cup\bar{z}_3)\cap(\bar{z}_1\cup z_2)$
5	$y(x_1,x_2,x_3)=(tgx_1)^{\sin x_2}+5^{x_3^2}$	$f(z_1,z_2,z_3)=\overline{\bar{z}_1\cap z_3}\cup z_2\cap\bar{z}_3\cap(\bar{z}_1\cup z_2)$
6	$y(x_1,x_2,x_3)=arctg\frac{x_1+x_2}{1+x_1x_2}+x_3$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap(z_2\cup\bar{z}_3)\cup\overline{(\bar{z}_1\cup z_2)}$
7	$y(x_1,x_2,x_3)=\arcsin\frac{2x_2}{\sqrt{x_1^2+x_3^2}}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_2)}\cap(z_2\cup\bar{z}_3)\cup\bar{z}_1\cap\bar{z}_3$
8	$y(x_1,x_2,x_3)=\sin\frac{x_1x_3^2}{x_1+x_2}+4^{x_1+2x_2-x_3}$	$f(z_1,z_2,z_3)=\overline{\overline{(\bar{z}_1\cup z_2)}\cap(\bar{z}_1\cup\bar{z}_3)}\cup\bar{z}_1\cap\bar{z}_3$
9	$y(x_1,x_2,x_3)=\lg\sqrt{\frac{x_1^2+x_2+1}{x_1x_3}}$	$f(z_1,z_2,z_3)=z_3\cap z_1\cup z_1\cap\bar{z}_3\cup\overline{(\bar{z}_2\cup z_3)}$

10	$y(x_1, x_2, x_3) = tg\dfrac{x_1 + x_2}{x_3^{\,2}} + 3^{x_1^{\,3} - 2x_1x_2 + x_3}$	$f(z_1, z_2, z_3) = (\bar{z}_3 \cup z_1) \cap \overline{(z_1 \cup \bar{z}_2)} \cup \bar{z}_2 \cap z_3$
11	$y(x_1, x_2, x_3) = \cos\dfrac{x_3}{(x_1 + x_2)^2} - x_3^{\,x_1 - x_2}$	$f(z_1, z_2, z_3) = \overline{\overline{\bar{z}_1 \cap z_2} \cup z_3} \cap \bar{z}_2 \cup (\bar{z}_1 \cup z_3)$
12	$y(x_1, x_2, x_3) = \sqrt[8]{9 + x_1^{\,2} + x_2} - \dfrac{x_2}{\sqrt{4x_1^{\,2} + x_3^{\,2}}}$	$f(z_1, z_2, z_3) = \overline{(\bar{z}_1 \cup z_2)} \cap \overline{(z_1} \cup z_3) \cup \overline{(\bar{z}_2 \cup z_3)}$
13	$y(x_1, x_2, x_3) = \dfrac{\sin(x_1x_2)}{x_1^{\,2} + x_2^{\,2}} - 3x_3^{\,x_1 + x_2}$	$f(z_1, z_2, z_3) = \overline{(z_1 \cup \bar{z}_3)} \cap (z_2 \cup \bar{z}_3) \cup \overline{\bar{z}_1 \cap \bar{z}_2}$
14	$y(x_1, x_2, x_3) = x_1^{\,\sin x_2} \cdot (x_1 + x_3^{\,2}) + x_1 \lg(x_3^{\,2} + 3x_2)$	$f(z_1, z_2, z_3) = \overline{\overline{(\bar{z}_2 \cup z_3)} \cap (\bar{z}_1 \cup \bar{z}_2)} \cup \bar{z}_1 \cap z_3$
15	$y(x_1, x_2, x_3) = \sqrt{\dfrac{x_1^{\,2} + x_2}{x_3^{\,2}}} + \lg x_1$	$f(z_1, z_2, z_3) = z_2 \cap z_3 \cup \bar{z}_1 \cap \bar{z}_3 \cup \overline{(z_1 \cup \bar{z}_2)}$
16	$y(x_1, x_2, x_3) = \dfrac{\lg(x_1 + x_2)}{\sqrt{x_1^{\,2} + 2x_2 + x_3}}$	$f(z_1, z_2, z_3) = \overline{(\bar{z}_2 \cup z_1)} \cap \overline{(z_2 \cup \bar{z}_3)} \cup \bar{z}_1 \cap z_3$
17	$y(x_1, x_2, x_3) = x_1^{\,x_2} \cdot \lg(x_1 + x_2 + x_3^{\,2})$	$f(z_1, z_2, z_3) = (z_1 \cup \bar{z}_3) \cap \overline{(z_1 \cup \bar{z}_2)} \cup \overline{(z_2 \cup \bar{z}_3)}$
18	$y(x_1, x_2, x_3) = \dfrac{\lg(x_1^{\,2} + x_2^{\,2} - 8)}{\sqrt{x_2^{\,2} + x_1^{\,2} - 4x_3}}$	$f(z_1, z_2, z_3) = \bar{z}_1 \cap z_2 \cup \overline{(z_2 \cup \bar{z}_3)} \cap \overline{(z_1 \cup \bar{z}_3)}$
19	$y(x_1, x_2, x_3) = \dfrac{2x_1 - 3x_2}{\lg(x_1 + 2)^2} - \sqrt{x_1 + x_2^{\,2} + 4x_3}$	$f(z_1, z_2, z_3) = \overline{(z_3 \cup z_2)} \cap (z_2 \cup \bar{z}_1) \cap (\bar{z}_1 \cup \bar{z}_3)$
20	$y(x_1, x_2, x_3) = \dfrac{x_1^{\,3} x_2^{\,3}}{x_1^{\,3} x_2^{\,3} + (x_1 + x_2)^4} - \sin x_3$	$f(z_1, z_2, z_3) = \overline{(\bar{z}_2 \cup z_3)} \cap \overline{(z_1 \cup \bar{z}_3)} \cap (\bar{z}_1 \cup z_2)$

Контрольні питання

1. Яке призначення має мова програмування ПЛК ST і коли вона застосовується?

2. Структура програми ST.

3. Вирази мови ST.

2. Що таке оператор в програмі ST і які види операторів бувають?

3. Яке призначення має оператор IF у програмі ST?

4. Яке призначення має оператор CASE в програмі ST?

5. Яким чином здійснюється виклик функції в програмі ST?

6. Яким чином здійснюється виклик функціонального блоку в програмі ST?

7. Які оператори циклу існують в мові ST, в чому їх особливості?

ЛАБОРАТОРНА РОБОТА № 4

Тема: ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ ISaGRAF НА МОВІ IL

Мета роботи: знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування IL.

Короткі теоретичні відомості

МОВА ІНСТРУКЦІЙ (IL)

Мова інструкцій (Instruction List, IL) - це текстова мова низького рівня. Інструкції завжди відносяться до **поточного результату (**або IL регістру). Оператор визначає операцію, яка має бути виконана з поточним результатом і операндом. Результат операції знову запам'ятовується в поточному результаті.

Основний синтаксис IL

IL програма - це список **інструкцій**. Кожна інструкція повинна розпочинатися з нового рядка і повинна містити **оператор**, з додатковим модифікаторами, якщо потрібно, для специфічних операцій, один або декілька **операндів**, розділених комою(,). Інструкції може передувати **мітка** з двокрапкою(:). Якщо до інструкції приєднаний **коментар**, то він повинен знаходитися у кінці рядка. Коментар завжди починається з (*і закінчується*). Між інструкціями може бути введений порожній рядок. Коментарі можуть бути поміщені в порожні рядки.

Нижче дані приклади рядків інструкцій:

```
Мітка Оператор  Операнд          Коментар
Start:  LD      IX1          (* натиснути кнопку *)
        ANDN    MX5          (* команда дозволена*)
        ST      QX2          (* запустити мотор *)
```

Мітки

Інструкції може передувати мітка з двокрапкою(:). Мітка може бути поміщена на порожній рядок. Мітки використовуються як операнди для деяких операцій, таких як **стрибки**. Імена міток повинні задовольняти наступним правилам:

- ім'я не може бути довше**16** символів;

- першим символом має бути **буква;**

- подальшими символами можуть бути **букви, цифри** або **символ підкреслення.**

У одній програмі одне і те ж ім'я не може бути використане для позначення більш ніж однієї мітки. Ім'я мітки може співпадати з ім'ям змінної.

Модифікатори оператора

Нижче представлені можливі модифікатори оператора. Символ модифікатора повинен завершувати ім'я оператора, без пропусків між ними.

N булеве заперечення операнда

(затримана операція

C умовна операція

Модифікатор **N** визначає булеве заперечення операнда. Наприклад, інструкція **ORN IX12** інтерпретується як: **result:= result OR NOT(IX12).**

Модифікатор дужка (вказує на те, що виконання інструкції має бути затримане до тих пір, поки не зустрінеться закрита дужка).

Модифікатор C вказує на те, що відповідна інструкція має бути виконана тільки якщо поточний результат має значення TRUE (відмінне від нуля).

Модифікатор **C**, може комбінуватися з модифікатором N, який вказує на те, що інструкція має бути виконана тільки якщо поточний результат має значення FALSE (0).

Оператори IL

Стандартні оператори мови IL представлені в таблиці 1.

Таблиця 1 – Стандартні оператори мови IL

Оператор	Модифікатор	Операнд	Опис
LD	N	змінна, константа	Завантажує операнд
ST	N	змінна	Запам'ятовує поточний результат
S		логічна змінна	Встановлює на TRUE
R		логічна змінна	Скидає на FALSE
CAL	C N	Ім'я екземпляра ФБ	Виклик функціонального блоку
JMP	C N	мітка	Стрибок на мітку
RET	C N		Повернення з підпрограми
)			Виконати затриману операцію
Функція			Виклик функції або підпрограми
AND	N (	логічний	логічне І
&	N (	логічний	логічне І
OR	N (	логічний	логічне АБО
XOR	N (	логічний	що виключає АБО
ADD	(	змінна,константа	Складання
SUB	(	змінна, константа	Віднімання
MUL	(	змінна, константа	Множення
DIV	(	змінна, константа	Ділення
GT	(	змінна, константа	Перевірити: >
GE	(	змінна, константа	Перевірити: >=
EQ	(	змінна, константа	Перевірити: =
LE	(	змінна, константа	Перевірити: <=
LT	(	змінна, константа	Перевірити: <
NE	(	змінна, константа	Перевірити: <

Приклад виконання роботи

Постановка задачі. Розробити та дослідити додаток на мові IL для віртуального контролера, який реалізує обчислення наступних арифметичних і логічних виразів:

$$y(x_1, x_2, x_3) = \sin\frac{x_3 x_1^{\,2}}{x_3 + x_2}$$

$$f(z_1, z_2, z_3) = \overline{\bar{z}_1 \cap z_2} \cup z_3 \cap (\bar{z}_1 \cup \bar{z}_3)$$

де x_1, x_2, x_3 – вхідні дійсні змінні; y – вихідна дійсна змінна; z_1, z_2, z_3 – вхідні булеві змінні; f – вихідна булева змінна.

Рішення задачі

1. Створити новий проєкт з ім'ям "project4" в системі ISaGRAF 3.4 .

2. Створити нову програму "program4". При виборі мови вказати мову IL (рисунок 1).

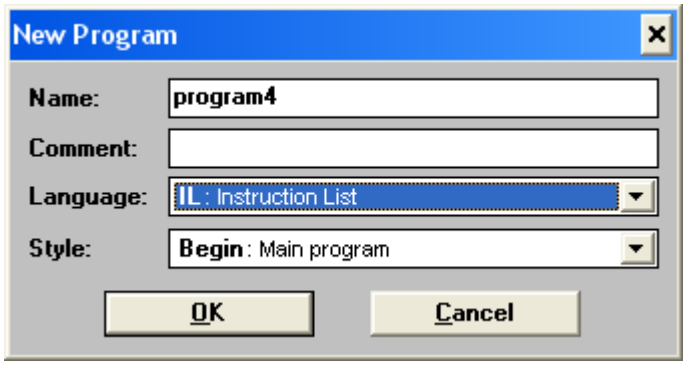


Рисунок 1 – Вибір мови IL

3. Оголосити використовувані змінні (рисунок 2).

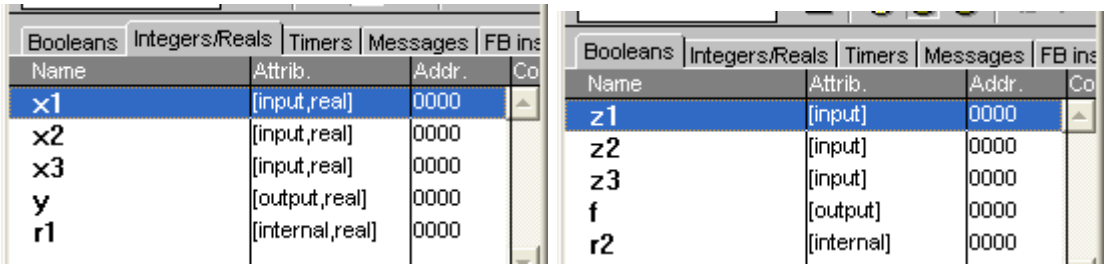


Рисунок 2 – Визначення змінних

4. Відредагувати програму відповідно до представленого нижче тексту (рисунок 3).

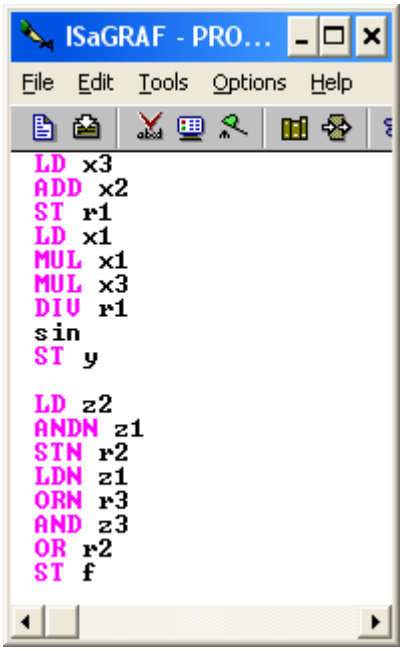


Рисунок 3 – Текст програми на мові IL

5. Настроїти конфігурацію введення/виведення і здійснити прив'язку вхідних і вихідних змінних проєкту.

6. Створити код додатка.

7. Провести налагодження додатка в режимі симуляції.

Виконання п.1 - 3, 5 - 7 детально розглянуто в лабораторній роботі № 1.

8. Виконати тестування і перевірку отриманих результатів для свого варіанту, як в лабораторній роботі № 1.

Тест № 1

Результати тестування приведені на рисунках 4 – 5.

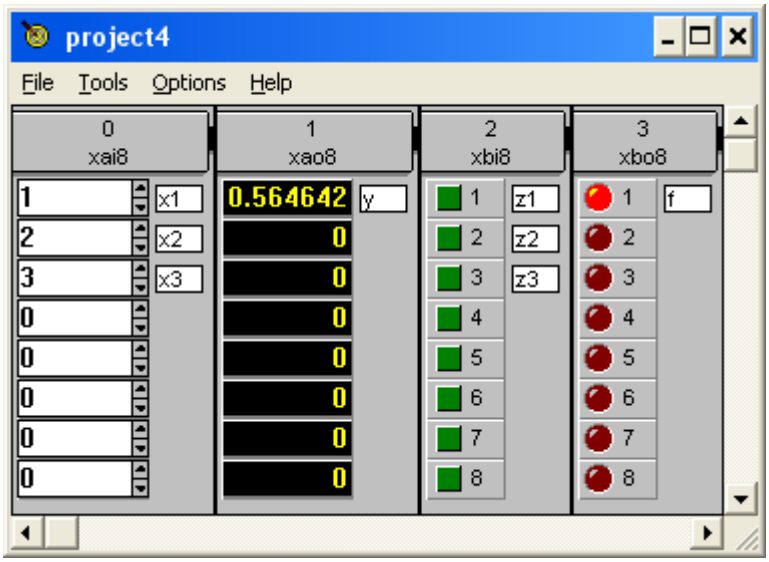
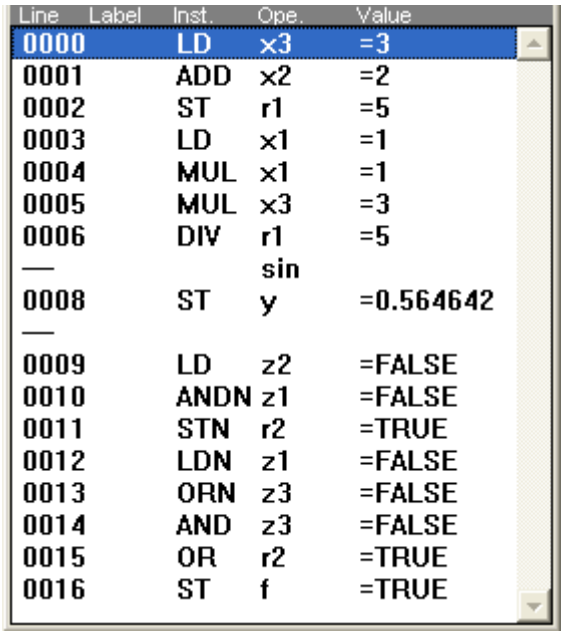


Рисунок 4 – Вікно симулятора з результатом тесту №1



Line	Label	Inst.	Ope.	Value
0000		LD	x3	=3
0001		ADD	x2	=2
0002		ST	r1	=5
0003		LD	x1	=1
0004		MUL	x1	=1
0005		MUL	x3	=3
0006		DIV	r1	=5
—			sin	
0008		ST	y	=0.564642
—				
0009		LD	z2	=FALSE
0010		ANDN	z1	=FALSE
0011		STN	r2	=TRUE
0012		LDN	z1	=FALSE
0013		ORN	z3	=FALSE
0014		AND	z3	=FALSE
0015		OR	r2	=TRUE
0016		ST	f	=TRUE

Рисунок 5 – Програма в режимі online

Тест № 2

Результати тестування приведені на рисунках 6 – 7.

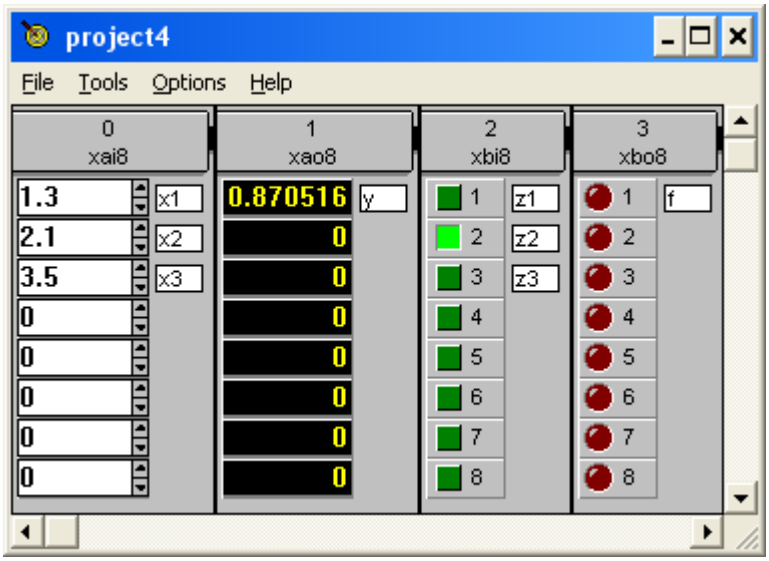


Рисунок 6 – Вікно симулятора з результатом тесту №2

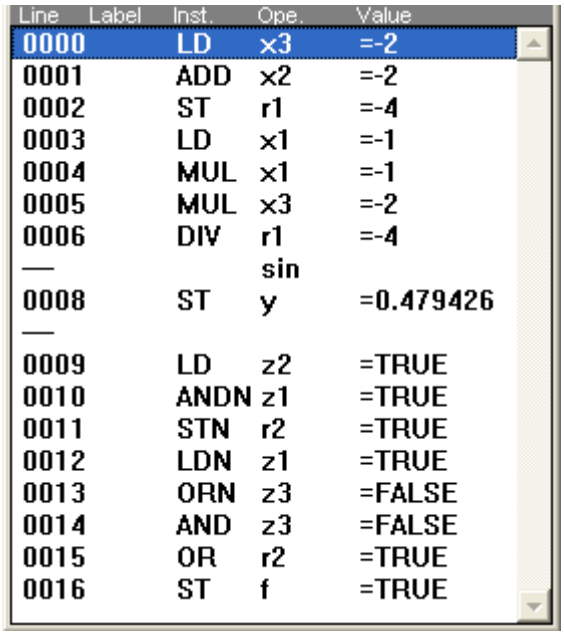


Рисунок 7 – Програма в режимі online

Тест № 3

Результати тестування приведені на рисунках 8 – 9.

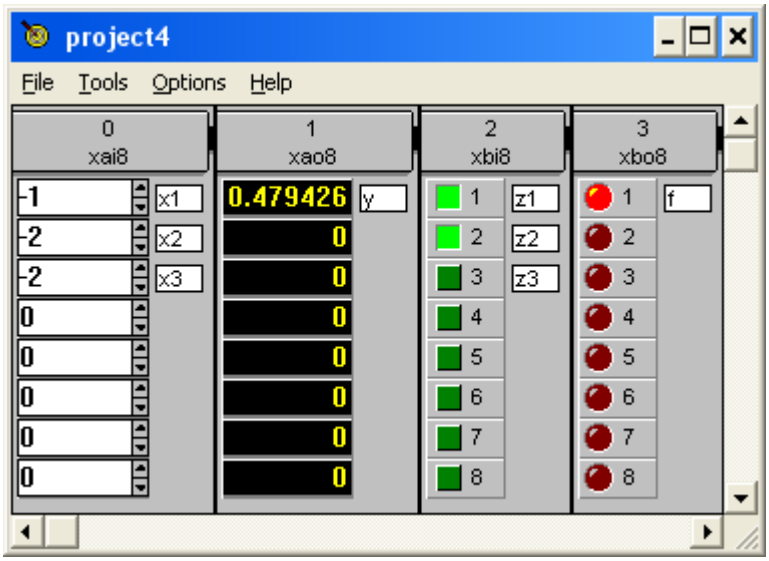


Рисунок 8 – Вікно симулятора з результатом тесту №3

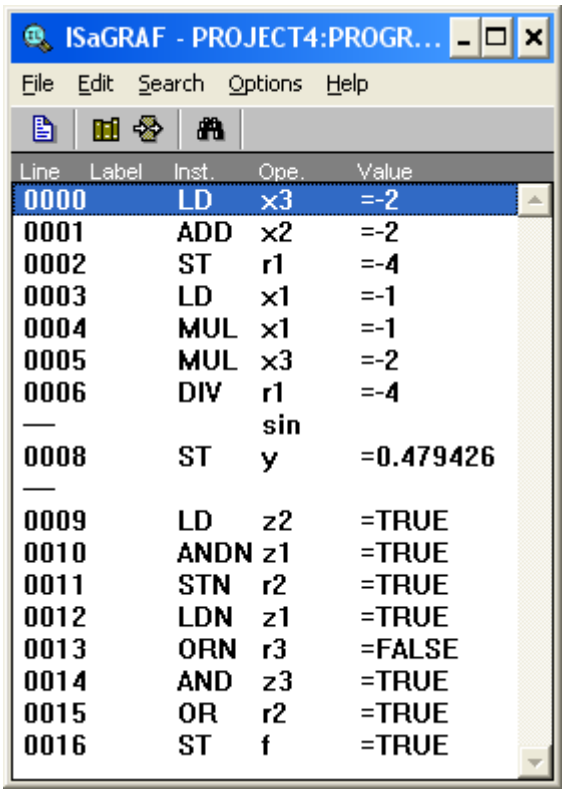


Рисунок 9 – Програма в режимі online

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 2.

Таблиця 2 – Варіанти завдань

№	Завдання 1	Завдання 2
1	$y(x_1,x_2,x_3)=\cos\frac{x_3x_1}{x_2}+5^{x_2^2+3x_1-x_3}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap(z_2\cup\bar{z}_3)\cup\overline{(\bar{z}_1\cup z_2)}$
2	$y(x_1,x_2,x_3)=\sin\frac{x_2x_1^2}{x_2+x_3}+3^{2x_2+x_3-x_1}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_2)}\cap(z_2\cup\bar{z}_3)\cup\bar{z}_1\cap\bar{z}_3$
3	$y(x_1,x_2,x_3)=\arccos\frac{x_3}{\sqrt{x_1^2+x_2^2}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)\cap(\bar{z}_1\cup\bar{z}_3)}\cup\bar{z}_1\cap\bar{z}_3$
4	$y(x_1,x_2,x_3)=(tgx_1)^{\sin x_2}+5^{x_3^2}$	$f(z_1,z_2,z_3)=z_3\cap z_1\cup z_1\cap\bar{z}_3\cup\overline{(\bar{z}_2\cup z_3)}$
5	$y(x_1,x_2,x_3)=arctg\frac{x_1+x_2}{1+x_1x_2}+x_3$	$f(z_1,z_2,z_3)=(\bar{z}_3\cup z_1)\cap\overline{(z_1\cup\bar{z}_2)}\cup\bar{z}_2\cap z_3$
6	$y(x_1,x_2,x_3)=\arcsin\frac{2x_2}{\sqrt{x_1^2+x_3^2}}$	$f(z_1,z_2,z_3)=(\bar{z}_2\cup z_3)\cap\overline{(z_1\cup\bar{z}_3)}\cup\overline{(\bar{z}_1\cup z_2)}$
7	$y(x_1,x_2,x_3)=\sin\frac{x_1x_3^2}{x_1+x_2}+4^{x_1+2x_2-x_3}$	$f(z_1,z_2,z_3)=z_2\cap\bar{z}_3\cup\overline{(\bar{z}_1\cup z_3)\cap(\bar{z}_1\cup\bar{z}_2)}$

8	$y(x_1,x_2,x_3)=\lg\sqrt{\frac{x_1^{\,2}+x_2+1}{x_1x_3}}$	$f(z_1,z_2,z_3)=\overline{(\overline{z_2}\cup z_1)}\cap(z_2\cup\bar{z}_3)\cap(\bar{z}_1\cup\bar{z}_3)$
9	$y(x_1,x_2,x_3)=tg\,\frac{x_1+x_2}{x_3^{\,2}}+3^{x_1^{\,3}-2x_1x_2+x_3}$	$f(z_1,z_2,z_3)=(\bar{z}_1\cup z_2)\cap\overline{(z_1\cup\bar{z}_3)}\cap\overline{(\bar{z}_2\cup z_3)}$
10	$y(x_1,x_2,x_3)=\cos\frac{x_3}{(x_1+x_2)^2}-x_3^{\,x_1-x_2}$	$f(z_1,z_2,z_3)=\overline{\overline{\bar{z}_1\cap z_2}\cup z_3}\cap\bar{z}_2\cup(\bar{z}_1\cup z_3)$
11	$y(x_1,x_2,x_3)=\sqrt[8]{9+x_1^{\,2}+x_2}-\frac{x_2}{\sqrt{4x_1^{\,2}+x_3^{\,2}}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_1\cup z_2)}\cap\overline{(\overline{z_1}\cup z_3)}\cup\overline{(\bar{z}_2\cup z_3)}$
12	$y(x_1,x_2,x_3)=\frac{\sin(x_1x_2)}{x_1^{\,2}+x_2^{\,2}}-3x_3^{\,x_1+x_2}$	$f(z_1,z_2,z_3)=\overline{(z_1\cup\bar{z}_3)}\cap(z_2\cup\bar{z}_3)\cup\overline{\bar{z}_1\cap\bar{z}_2}$
13	$y(x_1,x_2,x_3)=x_1^{\,\sin x_2}\cdot(x_1+x_3^{\,2})+x_1\lg(x_3^{\,2}+3x_2)$	$f(z_1,z_2,z_3)=\overline{\overline{(\bar{z}_2\cup z_3)}\cap(\bar{z}_1\cup\bar{z}_2)}\cup\bar{z}_1\cap z_3$
14	$y(x_1,x_2,x_3)=\sqrt{\frac{x_1^{\,2}+x_2}{x_3^{\,2}}}+\lg x_1$	$f(z_1,z_2,z_3)=z_2\cap z_3\cup\bar{z}_1\cap\bar{z}_3\cup\overline{(z_1\cup\bar{z}_2)}$
15	$y(x_1,x_2,x_3)=\frac{\lg(x_1+x_2)}{\sqrt{x_1^{\,2}+2x_2+x_3}}$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_2\cup z_1)}\cap\overline{(z_2\cup\bar{z}_3)}\cup\bar{z}_1\cap z_3$
16	$y(x_1,x_2,x_3)=x_1^{\,x_2}\cdot\lg(x_1+x_2+x_3^{\,2})$	$f(z_1,z_2,z_3)=(z_1\cup\bar{z}_3)\cap\overline{(z_1\cup\bar{z}_2)}\cup\overline{(z_2\cup\bar{z}_3)}$
17	$y(x_1,x_2,x_3)=\frac{\lg(x_1^{\,2}+x_2^{\,2}-8)}{\sqrt{x_2^{\,2}+x_1^{\,2}-4x_3}}$	$f(z_1,z_2,z_3)=\bar{z}_1\cap z_2\cup\overline{(z_2\cup\bar{z}_3)}\cap\overline{(z_1\cup\bar{z}_3)}$
18	$y(x_1,x_2,x_3)=\frac{2x_1-3x_2}{\lg(x_1+2)^2}-\sqrt{x_1+x_2^{\,2}+4x_3}$	$f(z_1,z_2,z_3)=\overline{(z_3\cup z_2)}\cap(z_2\cup\bar{z}_1)\cap(\bar{z}_1\cup\bar{z}_3)$
19	$y(x_1,x_2,x_3)=\frac{x_1^{\,3}x_2^{\,3}}{x_1^{\,3}x_2^{\,3}+(x_1+x_2)^4}-\sin x_3$	$f(z_1,z_2,z_3)=\overline{(\bar{z}_2\cup z_3)}\cap\overline{(z_1\cup\bar{z}_3)}\cap(\bar{z}_1\cup z_2)$
20	$y(x_1,x_2,x_3)=\frac{\sqrt{x_1^{\,2}+x_2^{\,2}}}{x_3^{\,2}}+\lg x_3$	$f(z_1,z_2,z_3)=\overline{\bar{z}_1\cap z_2}\cup\bar{z}_1\cap z_3\cap\overline{(z_2\cup\bar{z}_3)}$

Контрольні питання

1. Мова IL.
2. Структура програми IL.
3. Мітки IL програми.
4. Модифікатори IL оператора.
5. Оператори мови IL.

ЛАБОРАТОРНА РОБОТА № 5

Тема: ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ ISaGRAF НА МОВІ FC

Мета роботи: знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування FC.

Короткі теоретичні відомості

МОВА ПОТОКОВИХ ДІАГРАМ (FC)

Потокові діаграми (FlowChart, **FC)** - це графічна мова, що використовується для опису послідовних операцій. Потокова діаграма складається з **дій** і **тестів**. Між діями і тестами знаходяться **орієнтовані зв'язки**, що представляють потоки даних. Множинні зв'язки використовуються для представлення розбіжностей і сходжень. Дії і тести можуть бути описані за допомогою мов ST, LD або IL. Функції та функціональні блоки будь-якої мови (крім SFC) можуть бути викликані з дій і тестів. Програма потокових діаграм може викликати інші програми потокових діаграм.

Основні компоненти FC

Початок схеми FC

Символ "**Початок**" повинен з'являтися на початку програми потокових діаграм. Він унікальний і не може бути опущений. Він являє початковий стан діаграми, коли вона активізована. На рисунку 1 представлено графічне відображення символу "початок".

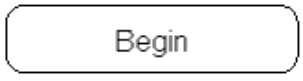


Рисунок 1 – Символ "Початок"

Символ "Початок" завжди має з'єднання (внизу) з іншим об'єктом схеми. Потокова діаграма невірна, якщо немає з'єднання символу "Початок" з іншим об'єктом.

Кінець схеми FC

Символ "**Кінець**" повинен виникати в кінці програми Потокових Діаграм. Він унікальний і не може бути пропущений. Може бути так, що ніякого з'єднання не підходить до символу "Кінець" (завжди виток), але символ "Кінець" все ж потрібний внизу схеми. Він являє собою заключний стан схеми, коли виконання було завершено. На рисунку 2 відображений символ "Кінець".

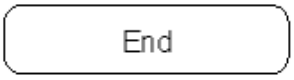


Рисунок 2 – Символ "Кінець"

Символ "Кінець" зазвичай має з'єднання (зверху) з іншими об'єктами схеми. Потокова діаграма може не мати з'єднання з об'єктом "Кінець" (нескінченний цикл). У цьому випадку об'єкт "Кінець" повинен бути внизу схеми.

Потокові зв'язки FC

Потокові зв'язки це лінії, які представляють потоки між двома точками в діаграмі. Зв'язок завжди закінчується стрілкою. На рисунку 3 приведено графічне відображення потокового зв'язку:

Рисунок 3 – Символ зв'язку

Два зв'язки не можуть виходити з одного джерела.

Дії FC

Символ **дії** являє собою дію, яку потрібно виконати. Дії ідентифікуються за допомогою числа і імені. На рисунку 4 приведено графічне відображення символу "Дії".

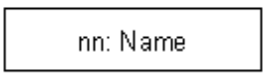


Рисунок 4 – Символ "Дія"

Два різні об'єкти однієї схеми не можуть мати одне і те ж ім'я або логічний номер. Мовами програмування для дій можуть бути ST, LD або IL. Дії завжди з'єднані зі зв'язками, одна підходить до нього, інша виходить з нього.

Умови FC

Умова є булевський тест. Умова ідентифікується числом і ім'ям. У відповідності зі значенням приєднаного виразу на ST, LD або IL, потік іде або по шляху "YES", або "NO". На рисунку 5 приведені можливі графічні відображення символу умови.

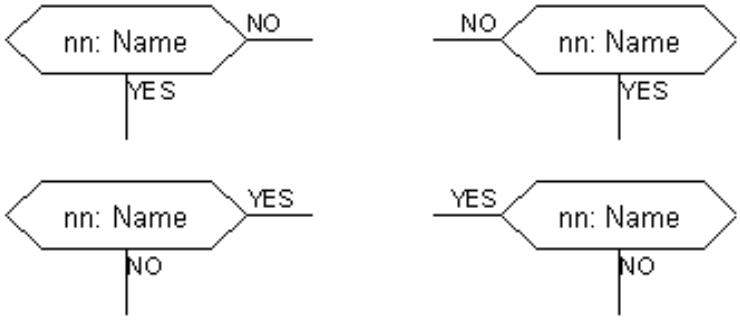


Рисунок 5 – Символ умови

Два різних об'єкта однієї і тієї ж схеми не можуть мати одне і теж ім'я або логічний номер.

Програма тесту - це:

- вираз на ST;

- одиночна сходинка в LD, з символом, приєднаним до унікального витка;

- кілька інструкцій на IL. Регістр IL (або поточний результат) використовується для того, щоб оцінити умову.

При програмуванні на ST, за виразом може слідувати двокрапка. При програмуванні на LD, значення умови представляється унікальним витком. Умова, яка дорівнює:

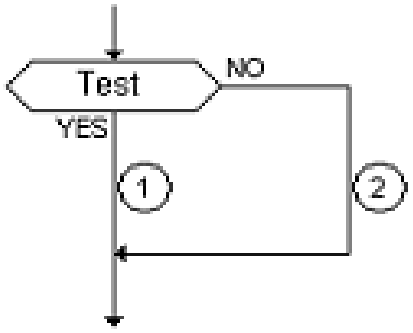
- 0 or FALSE направляє потік по NO

- 1 or TRUE направляє потік по YES.

Тест завжди з'єднаний з вхідним зв'язком, і обидва з'єднання, які виходять повинні бути визначені.

Приклади складних структур FC

На рисунках 6–8 показані приклади **складних структур**, які можуть бути визначені в потокових діаграмах. Такі структури об'єднують основні об'єкти зв'язані між собою.

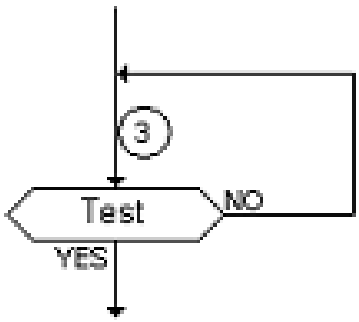


IF / THEN / ELSE

(1) місце для введення дій по "THEN"

(2) місце для введення дій по "ELSE"

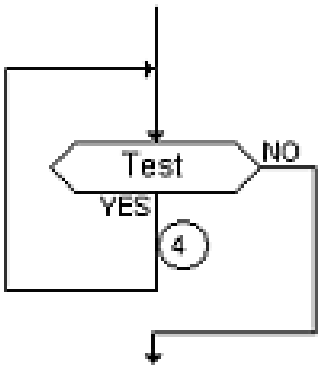
Рисунок 6 – Структура вибору



REPEAT / UNTIL

(3) місце для введення повторюючих дій

Рисунок 7 – Структура циклу До



WHILE / DO

(4) місце для введення повторюючих дій

Рисунок 8 – Структура циклу Доки

Приклад виконання роботи

Постановка задачі. Розробити та дослідити додаток на мові FC для віртуального контролера, який реалізує обчислення наступної функції:

$$y(x_1, x_2, x_3) = \begin{cases} x_1 - x_2, \text{ якщо } x_2 < x_3; \\ x_1 + x_2, \text{ якщо } x_2 > x_3; \\ x_1 x_2, \text{ якщо } x_2 = x_3 \end{cases}$$

де x_1, x_2, x_3 – вхідні цілі змінні; y – вихідна ціла змінна.

Рішення задачі

1. Створити новий проєкт з ім'ям "pz1" в системі ISaGRAF 3.4 .

2. Створити нову програму "prog_pz1". При виборі мови вказати мову FC (рисунок 9).

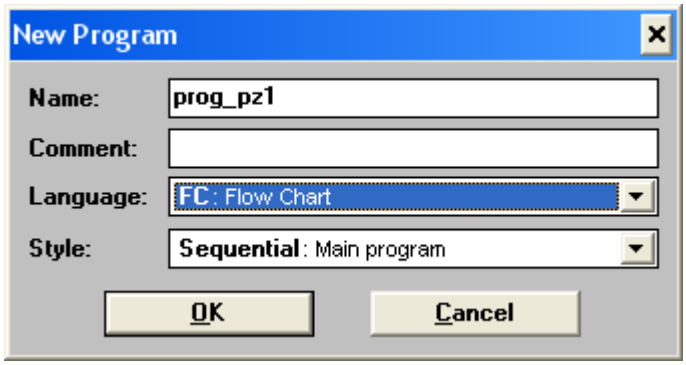


Рисунок 9 – Вікно створення FC-програми

3. Оголосити використовувані змінні (рисунок 10).

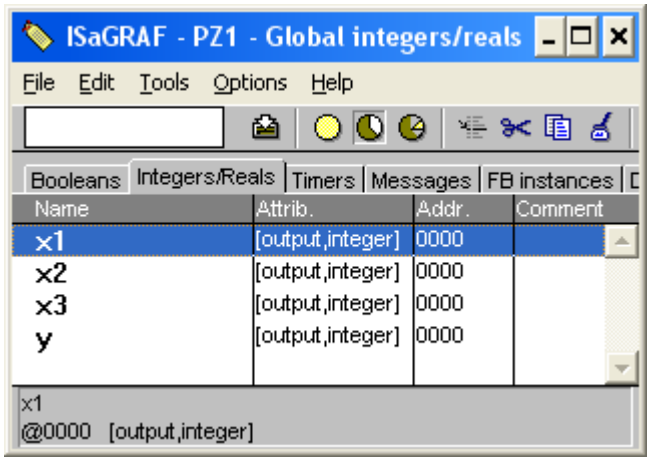


Рисунок 10 – Визначення змінних

4. Скласти програму відповідно до варіанта завдання.

Клацнути 1ЛКМ на кнопці **Insert an if-then-else**, підвести курсор до лінії з'єднання блоків **Begin** і **End**, ще раз натиснути 1ЛКМ. Побудувався тест з номером 03 (номера можуть бути іншими) (рисунок 11).

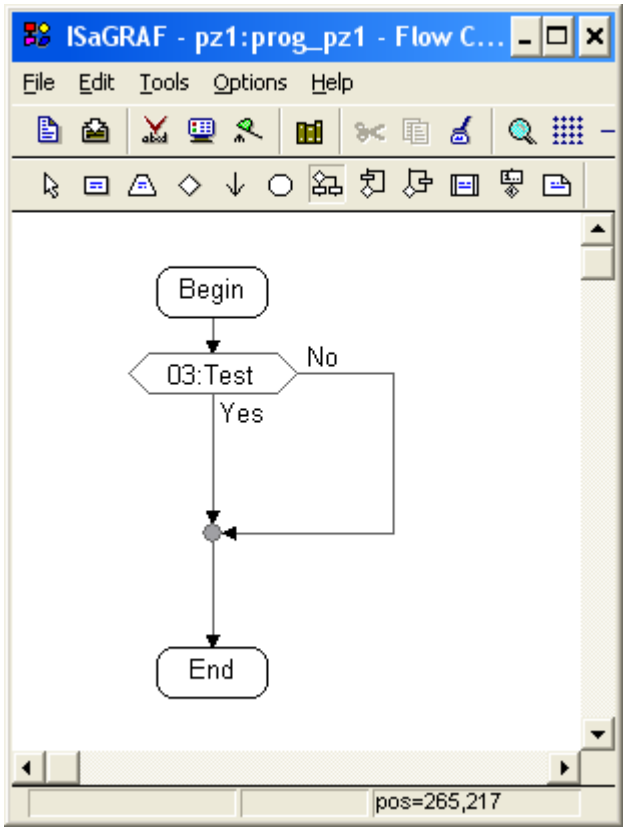


Рисунок 11 – Побудова початку програми

Аналогічно будуються ще два тести (07 і 09). За допомогою кнопки **Insert an action** розміщуються на діаграмі дії (**Action**), як показано на рисунку 12.

Необхідно описати всі дії і тести на мові ST.

Натиснути 2ЛКМ на **05: Action**. Задати початкові значення змінним x1, x2, x3 (рисунок 13).

В **03:Test** записати першу умову (рисунок 14).

У дії **06: Action** записати вираз функції для першої умови (рисунок 15).

Аналогічно запрограмувати другу умову і обчислення функції за цією умовою (рисунок 16).

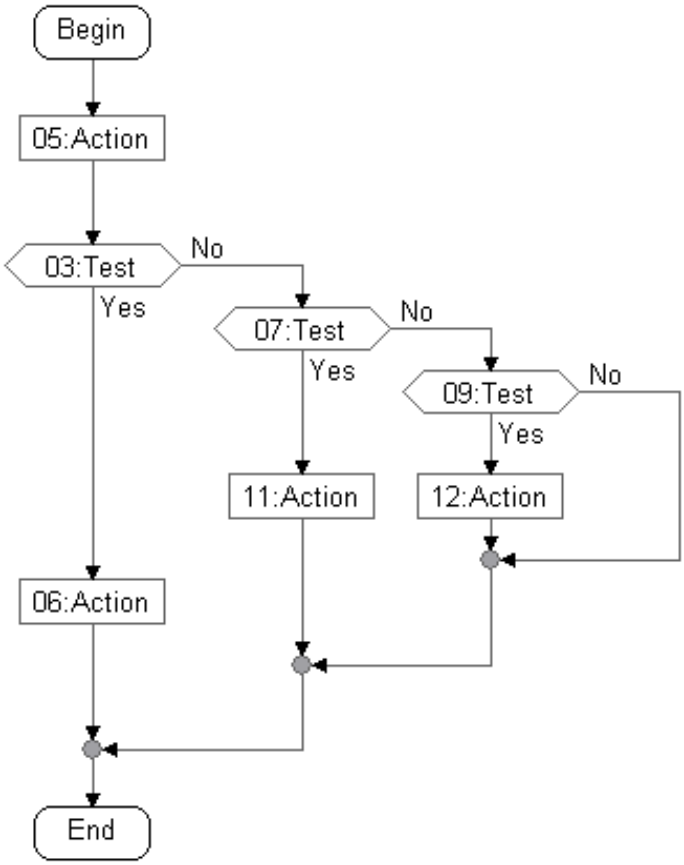


Рисунок 12 – Побудована FC-програма

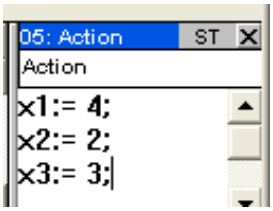


Рисунок 13 – Визначення початкової дії

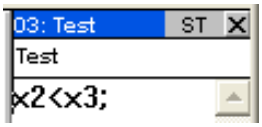


Рисунок 14 – Визначення першої умови

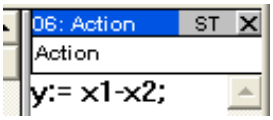


Рисунок 15 – Вираз функції для першої умови

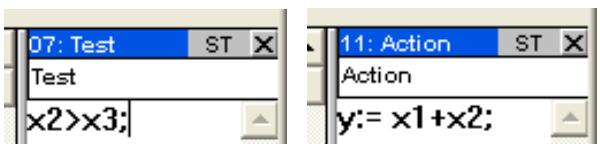


Рисунок 16 – Визначення другої умови і відповідної функції

Також запрограмувати третю умову і обчислення функції за цією умовою (рисунок 17).

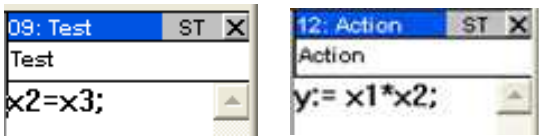


Рисунок 17 – Визначення третьої умови і відповідної функції

5. Настроїти конфігурацію введення/виведення і здійснити прив'язку вхідних і вихідних змінних проєкту (рисунок 18).

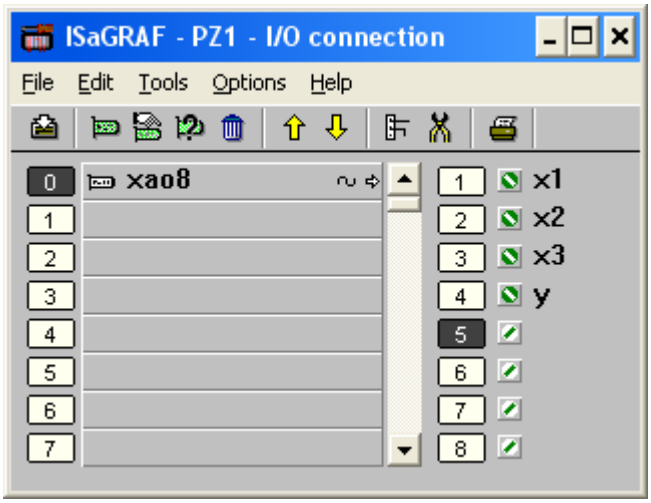


Рисунок 18 – Настроювання конфігурації введення/виведення

6. Створити код програми.

7. Провести налагодження додатка в режимі симуляції.

8. Виконати тестування програми (рисунок 19).

Тест № 1

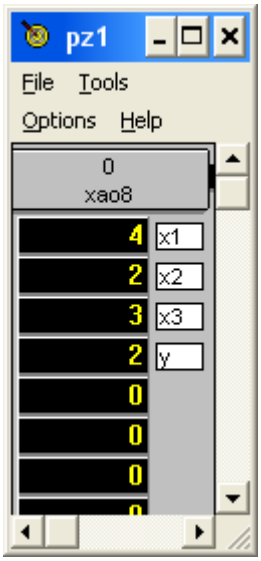


Рисунок 19 – Вікно симулятора з результатом тестування

Тест № 2

Змінити початкові значення змінним x1, x2, x3 (рисунок 20).

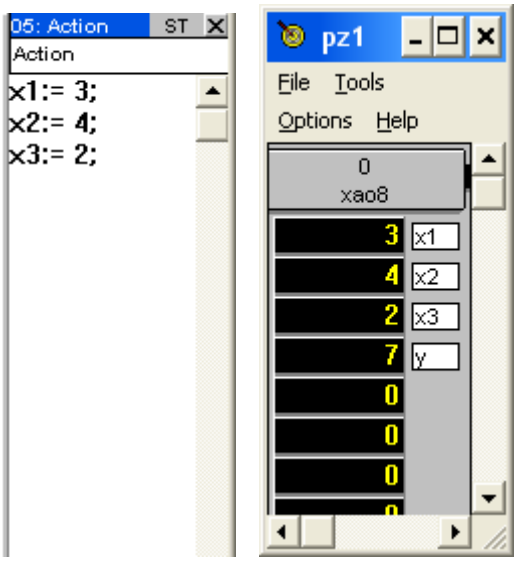


Рисунок 20 – Вікно симулятора з результатом тесту №2

Тест № 3

Змінити значення змінним x1, x2, x3 (рисунок 21).

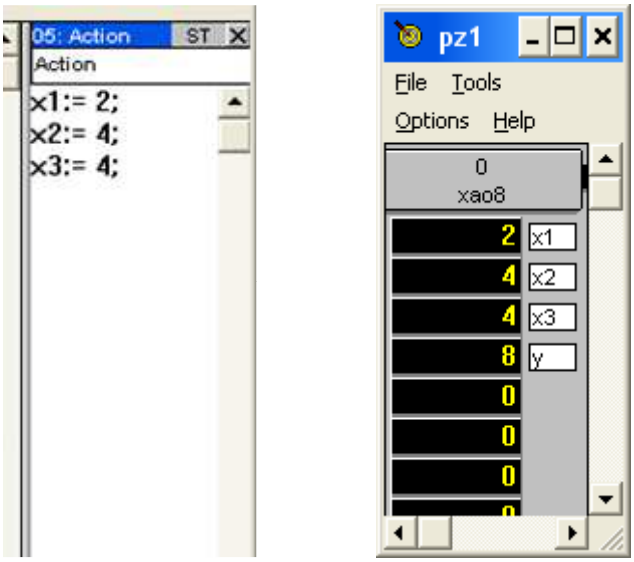


Рисунок 21 – Вікно симулятора з результатом тесту №3

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№	Вирази	№	Вирази
1	$y(x_1,x_2,x_3)=\begin{cases}x_1\cdot x_2,\ якщо\ x_3>x_1;\\ x_2^2-2,\ якщо\ x_3=x_1;\\ \sqrt{x_3},\ якщо\ x_3<x_1\end{cases}$	13	$y(x_1,x_2,x_3)=\begin{cases}{x_2}^2-x_1,\ якщо\ x_2-x_3<0;\\ {x_1}^2+3x_3,\ якщо\ x_2-x_3=0;\\ 10x_2,\ якщо\ x_2-x_3>0\end{cases}$
2	$y(x_1,x_2,x_3)=\begin{cases}{x_2}^2+6x_3,\ якщо\ x_1-x_3<2;\\ {x_2}^2-4x_3,\ якщо\ x_1-x_3=2;\\ 2tgx_2,\ якщо\ x_1-x_3>2\end{cases}$	14	$y(x_1,x_2,x_3)=\begin{cases}{x_1}^2,\ якщо\ x_3<0;\\ 3{x_2}^2,\ якщо\ x_3>0;\\ x_1-x_2,\ якщо\ x_3=0\end{cases}$
3	$y(x_1,x_2,x_3)=\begin{cases}-5x_2,\ якщо\ x_1+x_3<4;\\ 7-x_2,\ якщо\ x_1+x_3=4;\\ x_2+10,\ якщо\ x_1+x_3>4\end{cases}$	15	$y(x_1,x_2,x_3)=\begin{cases}2x_2,\ якщо\ x_1+x_2<1;\\ -x_3,\ якщо\ x_1+x_2=1;\\ x_1-1,\ якщо\ x_1+x_2>1\end{cases}$
4	$y(x_1,x_2,x_3)=\begin{cases}\ln x_2,\ якщо\ x_1x_3<8;\\ 1/x_2,\ якщо\ x_1x_3=8;\\ x_3+7,\ якщо\ x_1x_3>8\end{cases}$	16	$y(x_1,x_2,x_3)=\begin{cases}x_1-x_3,\ якщо\ x_1>0;\\ x_2+x_3,\ якщо\ x_1<0;\\ 0,\ якщо\ x_1=0\end{cases}$
5	$y(x_1,x_2,x_3)=\begin{cases}2(x_1-x_3),\ якщо\ x_2>3;\\ \sin(x_2+x_3),\ якщо\ x_2<3;\\ \lg x_1,\ якщо\ x_2=3\end{cases}$	17	$y(x_1,x_2,x_3)=\begin{cases}x_2x_3,\ якщо\ x_2+x_3<2;\\ -x_1x_3,\ якщо\ x_2+x_3=2;\\ x_1x_2,\ якщо\ x_2+x_3>2\end{cases}$
6	$y(x_1,x_2,x_3)=\begin{cases}2/{x_2}^2,\ якщо\ x_3-x_2<0;\\ 3x_3-x_1,\ якщо\ x_3-x_2=0;\\ x_1+4x_2,\ якщо\ x_3-x_2>0\end{cases}$	18	$y(x_1,x_2,x_3)=\begin{cases}3x_2,\ якщо\ x_1x_2<6;\\ 2x_3,\ якщо\ x_1x_2=6;\\ x_1+4,\ якщо\ x_1x_2>6\end{cases}$
7	$y(x_1,x_2,x_3)=\begin{cases}\cos x_2,\ якщо\ x_3-x_1<1;\\ 2x_1\cdot x_2,\ якщо\ x_3-x_1=1;\\ 4e^{x_2},\ якщо\ x_3-x_1>1\end{cases}$	19	$y(x_1,x_2,x_3)=\begin{cases}x_3-2,\ якщо\ x_1<x_2;\\ x_2+1,\ якщо\ x_1>x_2;\\ 2x_1,\ якщо\ x_1=x_2\end{cases}$
8	$y(x_1,x_2,x_3)=\begin{cases}x_3-x_1,\ якщо\ x_3<x_2;\\ x_2+12,\ якщо\ x_3=x_2;\\ x_2/x_1,\ якщо\ x_3>x_2\end{cases}$	20	$y(x_1,x_2,x_3)=\begin{cases}x_1-10,\ якщо\ x_2x_3<10;\\ 10x_2,\ якщо\ x_2x_3=10;\\ x_3+10,\ якщо\ x_2x_3>10\end{cases}$
9	$y(x_1,x_2,x_3)=\begin{cases}7x_2-2,\ якщо\ x_1/x_2<4;\\ 5\sin x_3,\ якщо\ x_1/x_2=4;\\ 2x_1+3,\ якщо\ x_1/x_2>4\end{cases}$	21	$y(x_1,x_2,x_3)=\begin{cases}2{x_2}^2,\ якщо\ x_1-x_2<1;\\ -3x_3,\ якщо\ x_1-x_2=1;\\ x_1+4x_2,\ якщо\ x_1-x_2>1\end{cases}$

10	$y(x_1, x_2, x_3) = \begin{cases} x_1 + 8, \text{ якщо } x_2 / x_3 < 3; \\ 8x_2, \text{ якщо } x_2 / x_3 = 3; \\ x_3 - 8, \text{ якщо } x_2 / x_3 > 3 \end{cases}$	22	$y(x_1, x_2, x_3) = \begin{cases} x_2^{\,2}, \text{ якщо } x_1 < x_3; \\ -(x_1 + x_2), \text{ якщо } x_1 > x_3; \\ x_1 + x_3, \text{ якщо } x_1 = x_3 \end{cases}$
11	$y(x_1, x_2, x_3) = \begin{cases} x_1 + 2x_2, \text{ якщо} x_3 - x_1 < 3; \\ \sqrt{x_1 \cdot x_2}, \text{ якщо} x_3 - x_1 > 3; \\ 5 / x_1 - x_2, \text{ якщо} x_3 - x_1 = 3 \end{cases}$	23	$y(x_1, x_2, x_3) = \begin{cases} 2x_1 + x_2, \text{ якщо } x_2 = 5; \\ \sqrt{x_1}, \text{ якщо } x_2 < 5; \\ \sin x_1, \text{ якщо } x_2 > 5 \end{cases}$
12	$y(x_1, x_2, x_3) = \begin{cases} \lg x_2, \text{ якщо } x_1 / x_3 < 5; \\ \sqrt{x_2 + 2}, \text{ якщо } x_1 / x_3 = 5; \\ x_3 - 9, \text{ якщо } x_1 / x_3 > 5 \end{cases}$	24	$y(x_1, x_2, x_3) = \begin{cases} 5x_2 \cdot x_3, \text{ якщо } x_1 > x_2; \\ x_3^2 + 1, \text{ якщо } x_1 = x_2; \\ \cos x_3, \text{ якщо } x_1 < x_2 \end{cases}$

Контрольні питання

1. Яке призначення має мова програмування ПЛК FC і коли вона застосовується?

2. Які основні компоненти мови FC?

3. Що представляють собою дії FC, якими мовами можна запрограмувати дії?

4. Що представляють собою умови FC, якими мовами можна запрограмувати умови?

5. Які можливі графічні відображення символу умови?

6. Яке призначення і відображення ліній потоку FC.

7. Яке призначення структури вибору?

8. Коли застосовуються структури циклу?

9. Чим структура циклу До відрізняється від структури циклу Доки?

ЛАБОРАТОРНА РОБОТА № 6

Тема: ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ ISaGRAF НА МОВІ SFC

Мета роботи: знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування SFC.

Короткі теоретичні відомості

МОВА ПОСЛІДОВНИХ ФУНКЦІОНАЛЬНИХ СХЕМ (SFC)

Мова послідовних функціональних схем SFC (Sequential Function Chart) – це графічна мова, що використовується для опису послідовних операцій. Процес представляється у вигляді набору певних кроків, пов'язаних переходами. До кожного переходу прикріплена логічна умова. Дії всередині кроків описані більш детально за допомогою інших мов (ST, IL, LD, FBD).

Основний формат схеми SFC

SFC програма – це графічний набір **кроків** і **переходів**, з'єднаних разом спрямованими зв'язками. Для позначення сходжень і розбіжностей використовуються множинні зв'язки. Деякі частини програми можуть бути відокремлені і представлені в основній схемі одним символом – **макрокроком**. Основні графічні правила для SFC наступні:

- кроки не можуть слідувати поспіль;
- переходи не можуть слідувати поспіль.

Основні компоненти SFC

Основні компоненти SFC: кроки, початкові кроки, переходи, орієнтовані зв'язки, стрибки на крок. Графічне позначення компонентів SFC має наступний вид:

Компонент	Назва компонента
	початковий крок
	крок
	перехід
	стрибок на крок
	макрокрок
	початковий макрокрок
	кінцевий макрокрок

Кроки та початкові кроки

Графічно крок представляється поодиноким **прямокутником**. Кожному кроку присвоюється **номер**, написаний всередині прямокутника. Основний опис кроку пишеться всередині прямокутника, приєднаного до символу кроку (рисунок 1). Це **вільний коментар** (який не являється частиною мови). Вищенаведена інформація називається **Рівнем 1** кроку.

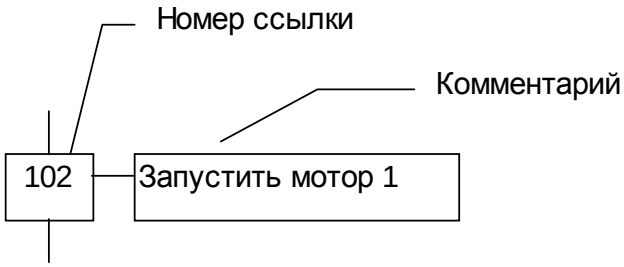


Рисунок 1 – Приклад кроку

Під час роботи **активний** крок помічається **маркером** (рисунок 2).

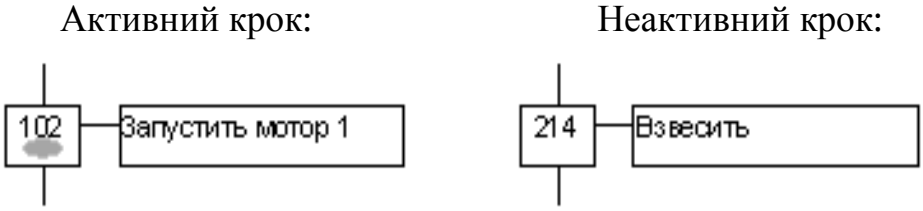


Рисунок 2 – Активний крок в роботі

Початкова ситуація програми SFC описується **початковими кроками**. Початковий крок позначається графічним символом з **подвійною рамкою**

(рисунок 3). Після запуску програми маркер автоматично встановлюється на кожен початковий крок.

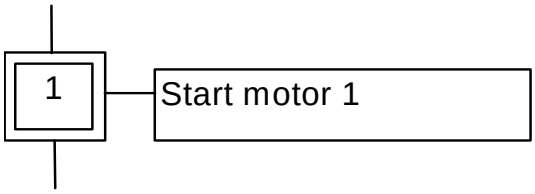


Рисунок 3 – Початковий крок

SFC програма повинна містити, принаймні, один початковий крок.

У кожного кроку є атрибути. Вони можуть бути використані в будь-якою іншою мовою в будь-якому місці програми:

GSnnn.x.....активність кроку (логічна змінна)

GSnnn.t.....тривалість активного стану кроку (таймер), де nnn - номер початкового кроку.

Переходи

Переходи представлені горизонтальними смужками, які перетинають лінії зв'язку. Кожному переходу присвоєно номер, наступний за символом переходу. Опис переходу розташовується праворуч від символу переходу (рисунок 4). Цей опис являє собою **вільний коментар** (що не входить в мову програмування). Вищенаведена інформація називається **Рівнем** 1 переходу.

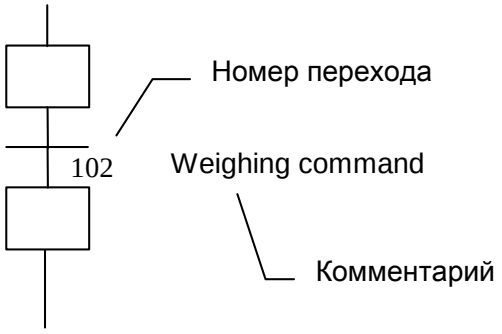


Рисунок 4 – Приклад переходу

Орієнтовані зв'язки

Для зв'язку кроків і переходів використовуються поодинокі лінії. Це орієнтовані зв'язки (рисунок 5). Зв'язки можуть бути зі стрілками (явна

орієнтація), або без стрілок (неявна орієнтація). Коли орієнтація не задана явно, зв'язок орієнтований зверху вниз.

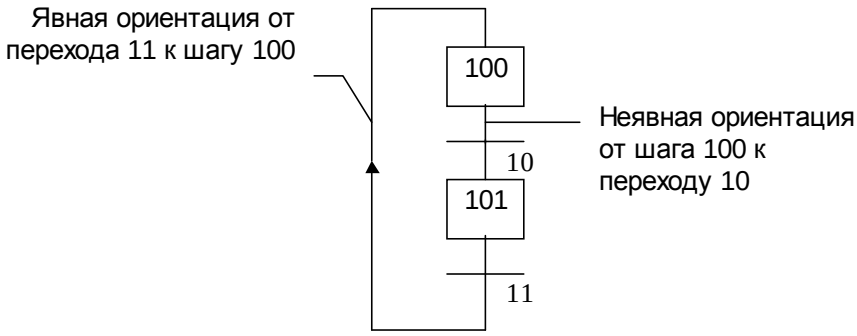


Рисунок 5 – Приклад орієнтованого зв'язку

Стрибок на крок

Символ стрибка може бути використаний, щоб визначити лінію зв'язку від переходу до кроку без рисування лінії. Символ стрибка повинен мати номер кроку призначення (рисунок 6).

Рисунок 6 – Приклад стрибка

Зв'язок від кроку до переходу можна представити за допомогою символу стрибка. Наступні схеми, які показані на рисунку 7 еквівалентні.

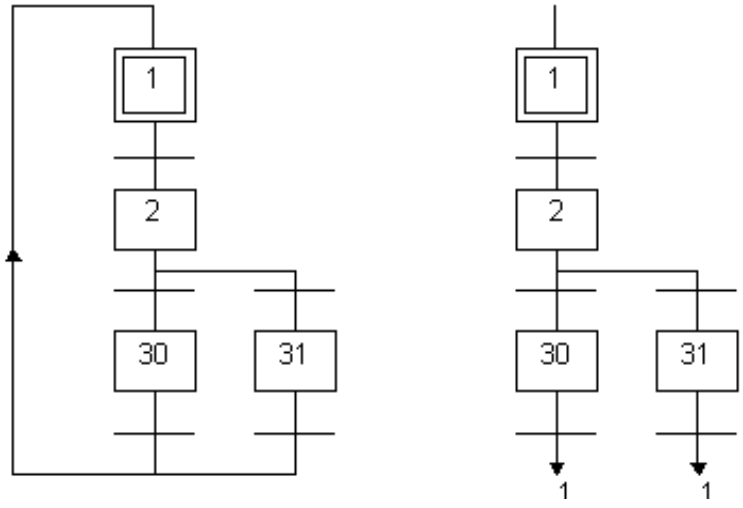


Рисунок 7 – Еквівалентні схеми

Сходження і розбіжності

Розбіжності - це множинні зв'язки від одного символу SFC (кроку або переходу) до багатьох. Сходження - це множинні зв'язки від більш ніж одного символу SFC до одного іншого символу. Сходження і розбіжності можуть бути поодинокими або подвійними.

Поодинокі розбіжності

Поодинока розбіжність - це множинний зв'язок від одного кроку до кількох переходів. Вона дозволяє маркеру активності пройти по одній з безлічі гілок. Поодиноке сходження - це множинний зв'язок від декількох переходів до одного і того ж кроку. Поодиноке сходження, зазвичай використовується для того, щоб об'єднати кілька гілок SFC, які почалися з поодинокої розбіжності. Поодинокі розбіжності і сходження позначаються поодинокими горизонтальними лініями (рисунок 8).

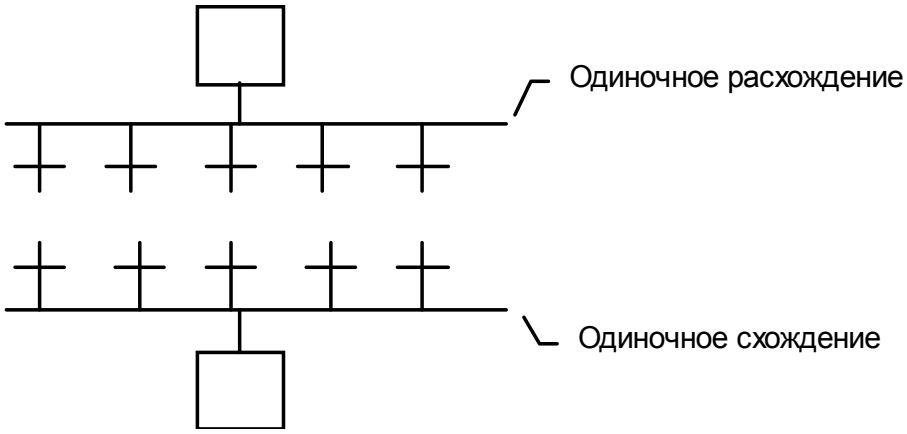


Рисунок 8 – Поодинокі розбіжності і сходження

Попередження: Умови, приєднані до переходів, не є **взаємовиключними**. Для того щоб програма пішла по одній гілці, треба явно визначити винятковість умов переходу (рисунок 9).

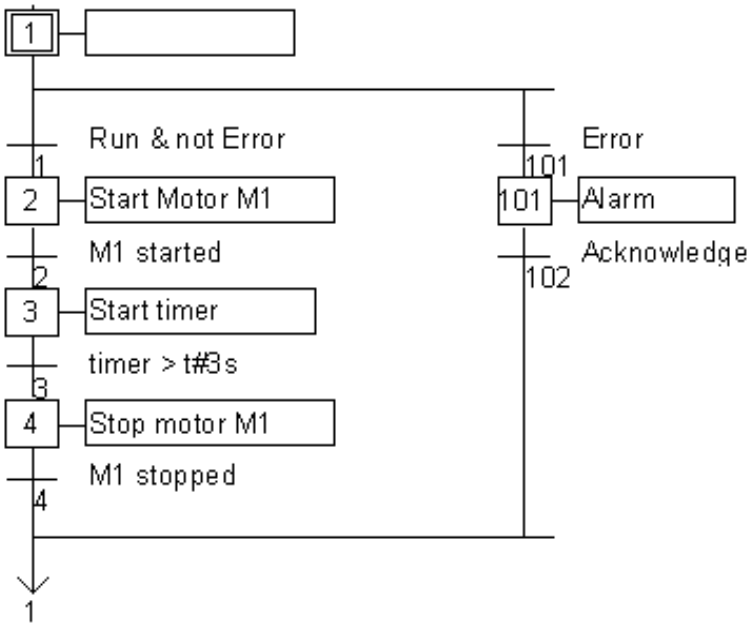


Рисунок 9 – Програма SFC з поодинокими сходженнями та розбіжностями

Подвійні розбіжності

Подвійна розбіжність - це множинний зв'язок від одного переходу до кількох кроків. Вона відповідає паралельній роботі процесу. Подвійне сходження - це множинний зв'язок від декількох кроків до одного і того ж переходу. Поодиноке сходження, зазвичай, використовується для того, щоб об'єднати кілька гілок SFC, що почалися з подвійного розбіжності. Подвійні розбіжності і сходження позначаються подвійними горизонтальними лініями (рисунок 10).

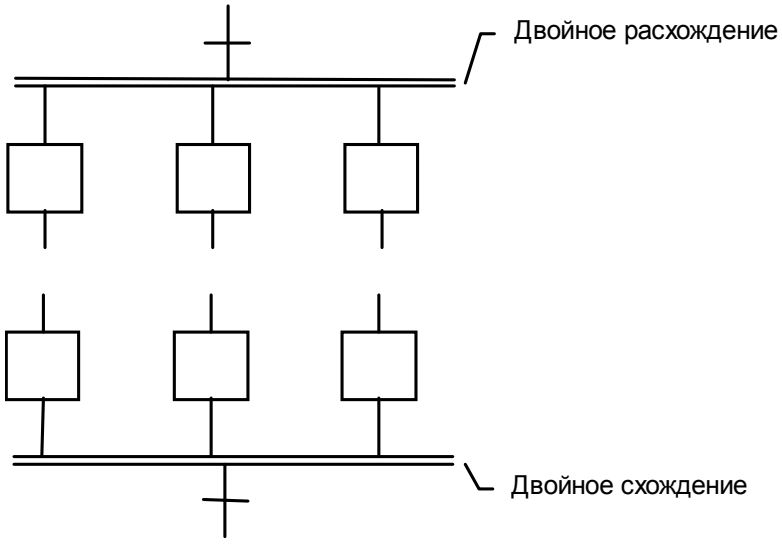


Рисунок 10 – Подвійні розбіжності і сходження

Приклад подвійного сходження і розбіжності приведений на рисунку 11.

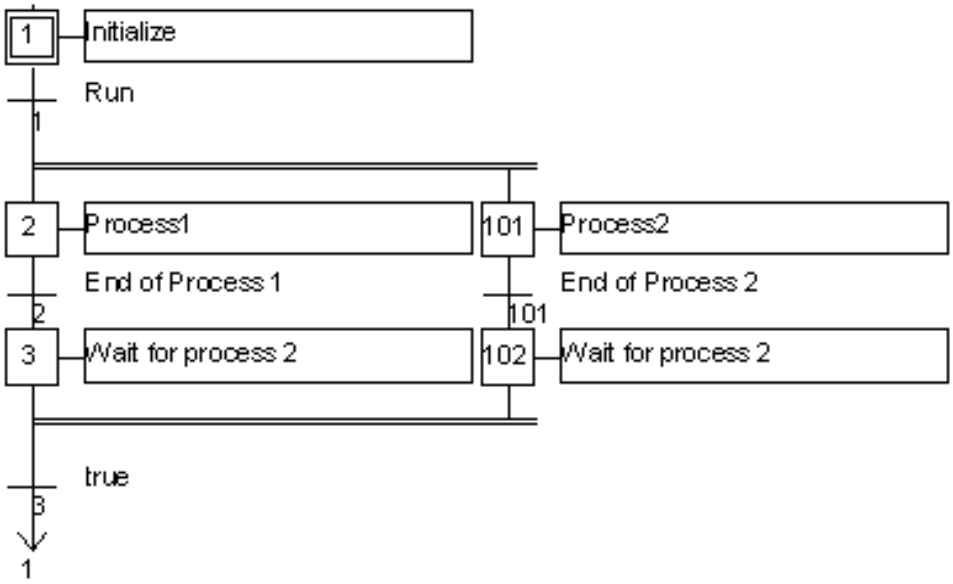


Рисунок 11 – Програма SFC з подвійними розбіжностями і сходженнями

Макрокроки

Макрокрок - це унікальне уявлення унікальної групи кроків і переходів. Тіло макрокроку описується окремо в іншому місці SFC програми. В основній схемі SFC він виглядає як один символ. Для макрокроку використовується символ, показаний на рисунку 12.

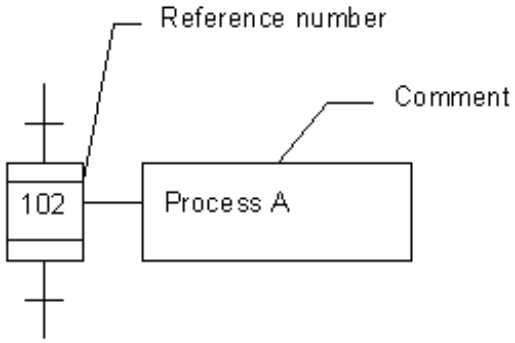


Рисунок 12 – Символ макрокроку

Номер, написаний в макрокроці, - це номер першого кроку в тілі макрокроку. Тіло макрокроку має починатися **початковим кроком** і закінчуватися **кінцевим кроком**. Схема повинна бути замкнутою. Початковий крок не має верхнього зв'язку (не має переходу позаду). Кінцевий крок не має нижнього зв'язку (не має переходу вперед). Символ макрокроку може бути поміщений в тіло іншого макрокроку (рисунок 13).

Попередження: Так як макрокрок це унікальний набір кроків і переходів, один і той же макрокрок не може бути використаний більш ніж один раз в SFC програмі.

(* Основна схема *) (* Тіло макрокроку *)

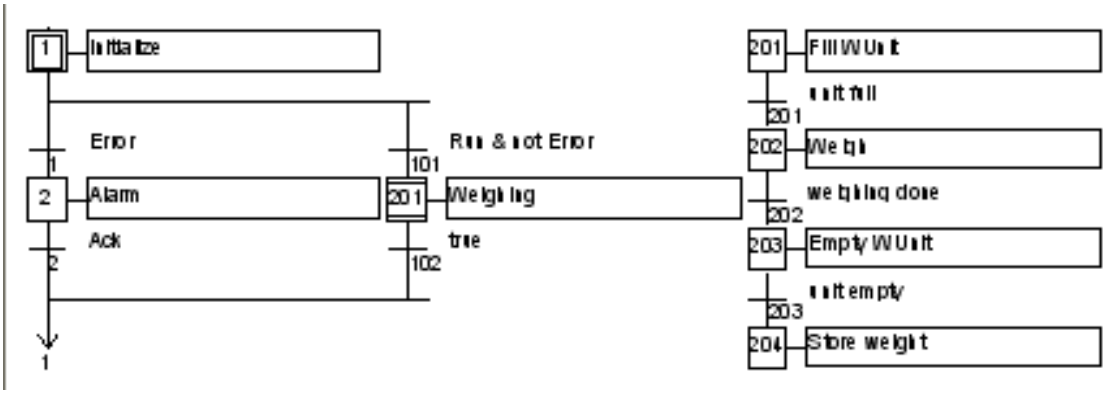


Рисунок 13 – Приклад програми SFC с макрокроком

Дії всередині кроків

Рівень 2 кроку SFC представляє собою детальний опис дій в період активності кроку. Цей опис може використовувати текстові доповнення мови SFC і інші мови, такі як структурний текст (ST).

Основні типи дій: булевські дії; імпульсні дії, описані на ST (дії типу "P"); дії, які не зберігаються, описані на ST (дії типу "N"); SFC дії.

Булевські дії

Булевські дії привласнюють значення логічній змінній при активізації кроку. Логічні змінні можуть бути вихідними або внутрішніми. Їм присвоюється значення кожен раз, коли крок стає активним або перестає бути активним. Синтаксис основних логічних дій:

<boolean_variable> (N) ; присвоїти змінній сигнал активності кроку;

<boolean_variable> ; той же ефект (N не обов'язково);

/<boolean_variable> ; присвоїти змінній заперечення сигналу активності кроку.

Є також можливості установки і скидання логічних змінних, коли крок стає активним. Синтаксис установки і скидання логічних дій:

<boolean_variable> (S); присвоює змінній значення TRUE, коли крок стає активним;

<boolean_variable> (R); присвоює змінній значення FALSE, коли крок стає активним.

Логічні змінні повинні бути вихідними (OUTPUT) або внутрішніми (INTERNAL). Наступна програма веде себе таким чином (рисунок 14).

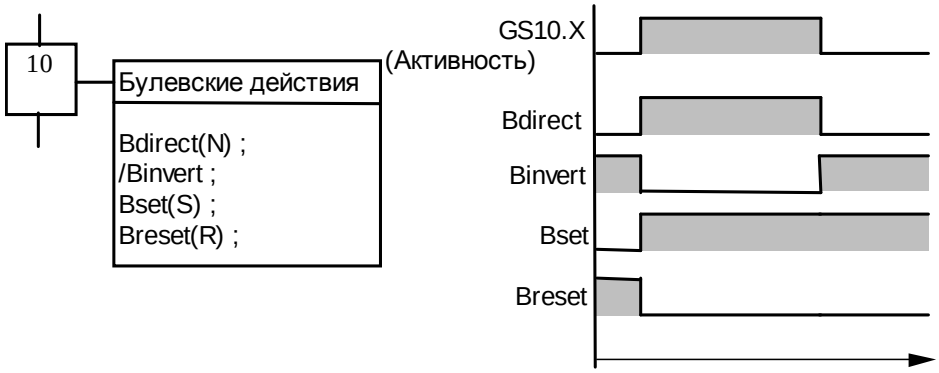


Рисунок 14 – Приклад логічних змінних

Приклад програми SFC, яка використовує булевські дії, приведений на рисунку 15.

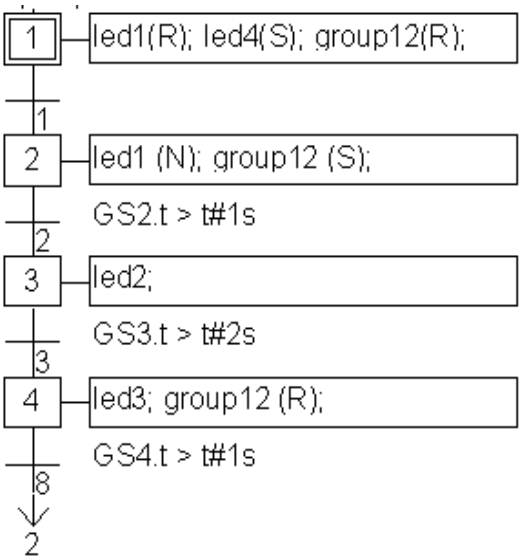


Рисунок 15 – Приклад програми SFC, яка використовує булевські дії

Декілька дій однакових або різних типів можуть бути описані в одному кроці. Засоби, що дозволяють використовувати будь-яку іншу мову:

- Виклик функцій і функціональних блоків з дії;
- Угоди мови IL.

Імпульсні дії

Імпульсна дія - це список інструкцій ST або IL, які виконуються лише один раз при активізації кроку. Інструкції пишуться у відповідності з наступним синтаксисом:

ACTION (P) :

(*ST оператори *)

END_ACTION ;

Результат імпульсної дії, приведений на рисунку 16.

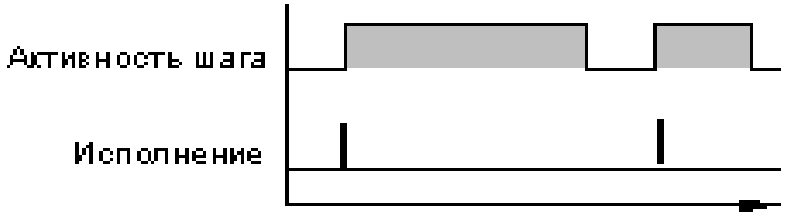


Рисунок 16 – Результат імпульсної дії

Приклад використання імпульсних дій приведений на рисунку 17.

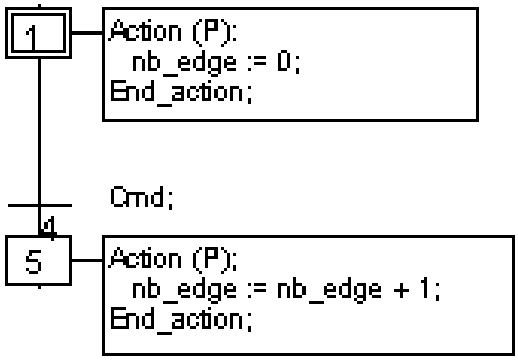


Рисунок 17 – Приклад використання імпульсних дій

Дії, що не зберігаються

Дія, що не зберігається - це список інструкцій ST або IL, які виконуються на кожному циклі, протягом усього періоду активності кроку. Інструкції пишуться у відповідності з наступним синтаксисом:

ACTION (N) :

(* ST оператори *)

END_ACTION ;

Результат дії, яка не зберігається, приведений на рисунку 18.

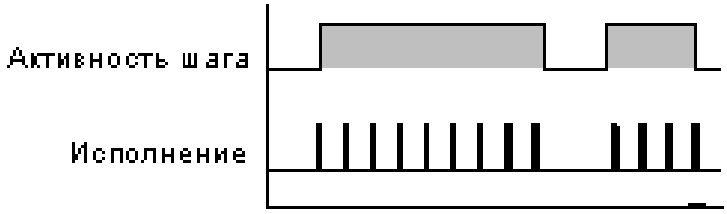


Рисунок 18. Результат дії, яка не зберігається

Приклад дії, яка не зберігається, приведений на рисунку 19.

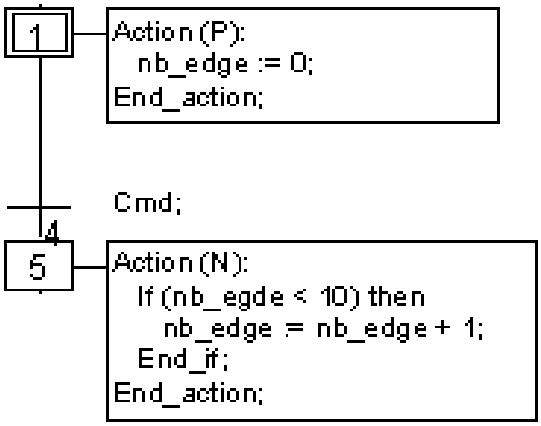


Рисунок 19 – Приклад дії, яка не зберігається

Дії SFC

SFC дія - це дочірня послідовність SFC, яка стартує і вбиває відповідно до зміни сигналу активності кроку. SFC дія може мати ознаку N (яка, не запам'ятовується), R (встановити), S (скинути). Ось синтаксис основних SFC дій:

<child_prog> (N); запустити дочірню послідовність, коли крок стає активним і вбити її, коли крок стає пасивним;

<child_prog> ; той же ефект (N не обов'язково);

<child_prog> (S); запустити дочірню послідовність, коли крок стає активним і нічого не робити, коли крок стає пасивним%

<child_prog> (R); вбити дочірню послідовність, коли крок стає активним і нічого не робити, коли крок стає пасивним.

SFC послідовність, визначена як дія, повинна бути дочірньою SFC програмою редагованої програми. Зауважимо, що використання ознак S і R для

SFC дії має той же ефект, що і оператори GSTART і GKILL в імпульсній дії на мові ST.

Нижче представлений приклад SFC дії. Основна SFC програма називається Father. Вона має два SFC спадкоємця - SeqMix і SeqPump. Текст батьківської SFC програми приведений на рисунку 20.

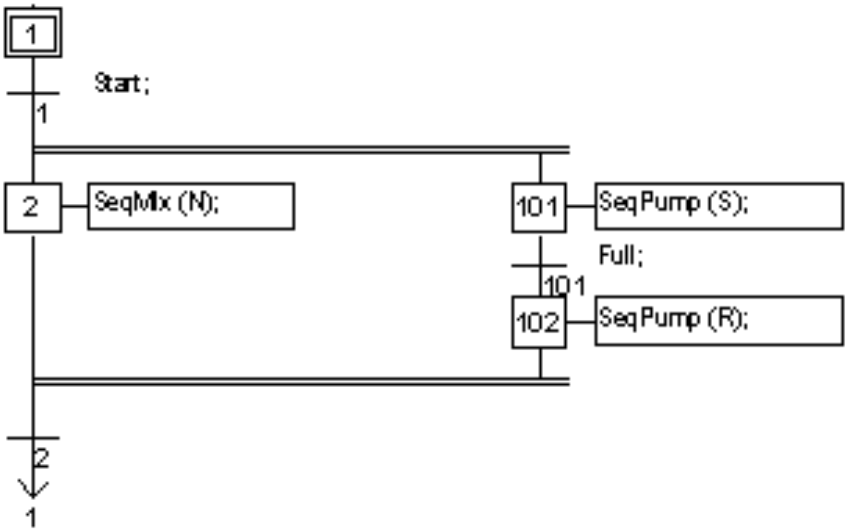


Рисунок 20 – Програма SFC, яка використовує дії SFC

Приклад виконання роботи

Постановка задачі. Розробити та дослідити додаток на мові SFC для віртуального контролера, який реалізує обчислення наступної функції:

$$y(x_1, x_2, x_3) = \begin{cases} x_1 - x_2, \ якщо \ x_2 < x_3; \\ x_1 + x_2, \ якщо \ x_2 > x_3; \\ x_1 x_2, \ якщо \ x_2 = x_3 \end{cases},$$

де x_1, x_2, x_3 - вхідні цілі змінні; y - вихідна ціла змінна.

Рішення задачі

1. Виконати загрузку системи ISaGRAF: Пуск → Програми → ISaGRAF 3.4 → Projects.

2. Створити новий проєкт з ім'ям "pz6": File → New (кнопка Create new project) у вікні Project Management.

3. Відкрити проєкт: 2ЛКМ на імені проєкту pz6 (кнопка Open) вікна Project Management.

4. Створити нову програму: File → New (кнопка Create new program) у вікні Programs.

• Введіть ім'я програми "prog6".

• Виберіть мову "**SFC**: Sequential Function Chart".

• Виберіть стиль "**Sequential:** Main program".

• Натисніть кнопку "Ok".

5. Оголосити використовувані змінні: File → Dictionary (кнопка). Виберіть закладку Integers/Reals, створіть аналогові змінні: x1 – x3 (internal, integer), y (output, integer).

6. Виконати редагування тексту програми: 2ЛКМ на імені програми prog6 (кнопка Edit program вікна Programs). Створити програму на мові SFC (рисунок 21).

7. Виконати програмування кроків на мові ST (рисунок 22).

8. Виконати програмування переходів на мові ST (рисунок 23).

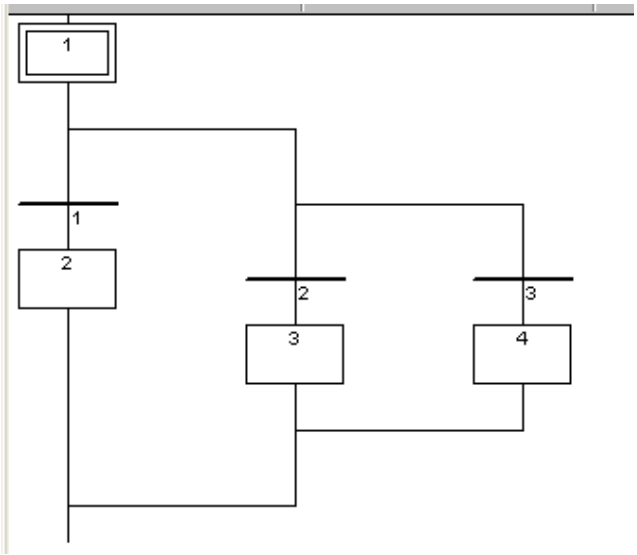


Рисунок 21 – Програма на мові SFC

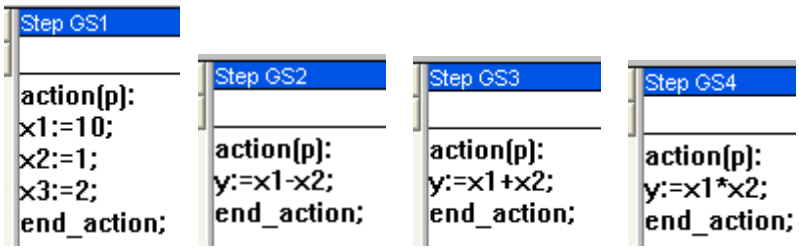


Рисунок 22 – Програмування кроків

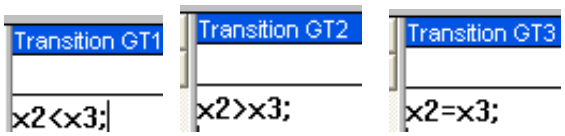


Рисунок 23 – Програмування переходів

9. Настроїти конфігурацію введення/виведення: Project → I/O connection вікна Programs (кнопка). Виберіть віртуальні плати введення/виведення **xai8**, **xao8**, **xbi8**, **xbo8**.

10. Здійснити прив'язку змінних введення/виведення: Виділіть рядок плати **xai8** і в меню "Edit" виберіть команду "Set channel/parameter".

11. Створити код програми: Make → Make application вікна Programs (кнопка Make application code). Якщо помилок немає, то натиснути Exit.

12. Виконати симуляцію: Debug → Simulate вікна Programs (кнопка). В меню "Options" вікна симулятора відзначте команду "Variable names".

13. Виконати тестування програми (рисунок 24).

Тест № 1

Рисунок 24 – Результат тестування програми

Тест № 2

Змініть початкові значення змінним x1, x2, x3 (рисунок 25).

Рисунок 25 – Вікно симулятора з результатом тесту №2

Тест № 3

Змініть значення змінним x1, x2, x3 (рисунок 26).

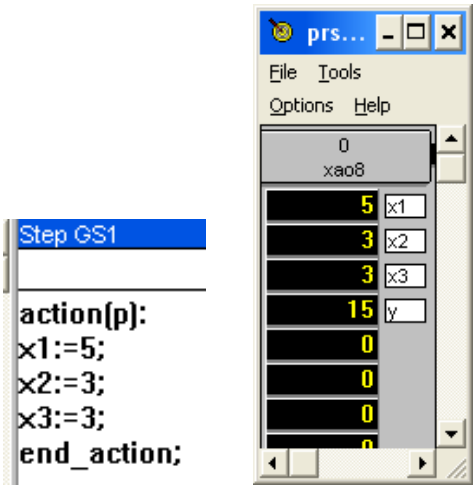


Рисунок 26 – Вікно симулятора з результатом тесту №3

Варіанти завдань

Варіанти завдань, необхідних для виконання даної лабораторної роботи, наведені в таблиці 1.

Таблиця 1 – Варіанти завдань

№	Вирази	№	Вирази
1	$y(x_1,x_2,x_3)=\begin{cases} {x_2}^2-x_1, \text{ якщо } x_2-x_3<0;\\ {x_1}^2+3x_3, \text{ якщо } x_2-x_3=0;\\ 10x_2, \text{ якщо } x_2-x_3>0 \end{cases}$	13	$y(x_1,x_2,x_3)=\begin{cases} x_1\cdot x_2, \text{ якщо } x_3>x_1;\\ x_2^2-2, \text{ якщо } x_3=x_1;\\ \sqrt{x_3}, \text{ якщо } x_3<x_1 \end{cases}$
2	$y(x_1,x_2,x_3)=\begin{cases} {x_1}^2, \text{ якщо } x_3<0;\\ 3{x_2}^2, \text{ якщо } x_3>0;\\ x_1-x_2, \text{ якщо } x_3=0 \end{cases}$	14	$y(x_1,x_2,x_3)=\begin{cases} {x_2}^2+6x_3, \text{ якщо } x_1-x_3<2;\\ {x_2}^2-4x_3, \text{ якщо } x_1-x_3=2;\\ 2tgx_2, \text{ якщо } x_1-x_3>2 \end{cases}$
3	$y(x_1,x_2,x_3)=\begin{cases} 2x_2, \text{ якщо } x_1+x_2<1;\\ -x_3, \text{ якщо } x_1+x_2=1;\\ x_1-1, \text{ якщо } x_1+x_2>1 \end{cases}$	15	$y(x_1,x_2,x_3)=\begin{cases} -5x_2, \text{ якщо } x_1+x_3<4;\\ 7-x_2, \text{ якщо } x_1+x_3=4;\\ x_2+10, \text{ якщо } x_1+x_3>4 \end{cases}$
4	$y(x_1,x_2,x_3)=\begin{cases} x_1-x_3, \text{ якщо } x_1>0;\\ x_2+x_3, \text{ якщо } x_1<0;\\ 0, \text{ якщо } x_1=0 \end{cases}$	16	$y(x_1,x_2,x_3)=\begin{cases} \ln x_2, \text{ якщо } x_1x_3<8;\\ 1/x_2, \text{ якщо } x_1x_3=8;\\ x_3+7, \text{ якщо } x_1x_3>8 \end{cases}$
5	$y(x_1,x_2,x_3)=\begin{cases} x_2x_3, \text{ якщо } x_2+x_3<2;\\ -x_1x_3, \text{ якщо } x_2+x_3=2;\\ x_1x_2, \text{ якщо } x_2+x_3>2 \end{cases}$	17	$y(x_1,x_2,x_3)=\begin{cases} 2(x_1-x_3), \text{ якщо } x_2>3;\\ \sin(x_2+x_3), \text{ якщо } x_2<3;\\ \lg x_1, \text{ якщо } x_2=3 \end{cases}$
6	$y(x_1,x_2,x_3)=\begin{cases} 3x_2, \text{ якщо } x_1x_2<6;\\ 2x_3, \text{ якщо } x_1x_2=6;\\ x_1+4, \text{ якщо } x_1x_2>6 \end{cases}$	18	$y(x_1,x_2,x_3)=\begin{cases} 2/{x_2}^2, \text{ якщо } x_3-x_2<0;\\ 3x_3-x_1, \text{ якщо } x_3-x_2=0;\\ x_1+4x_2, \text{ якщо } x_3-x_2>0 \end{cases}$
7	$y(x_1,x_2,x_3)=\begin{cases} x_3-2, \text{ якщо } x_1<x_2;\\ x_2+1, \text{ якщо } x_1>x_2;\\ 2x_1, \text{ якщо } x_1=x_2 \end{cases}$	19	$y(x_1,x_2,x_3)=\begin{cases} \cos x_2, \text{ якщо } x_3-x_1<1;\\ 2x_1\cdot x_2, \text{ якщо } x_3-x_1=1;\\ 4e^{x_2}, \text{ якщо } x_3-x_1>1 \end{cases}$
8	$y(x_1,x_2,x_3)=\begin{cases} x_1-10, \text{ якщо } x_2x_3<10;\\ 10x_2, \text{ якщо } x_2x_3=10;\\ x_3+10, \text{ якщо } x_2x_3>10 \end{cases}$	20	$y(x_1,x_2,x_3)=\begin{cases} x_3-x_1, \text{ якщо } x_3<x_2;\\ x_2+12, \text{ якщо } x_3=x_2;\\ x_2/x_1, \text{ якщо } x_3>x_2 \end{cases}$
9	$y(x_1,x_2,x_3)=\begin{cases} 2{x_2}^2, \text{ якщо } x_1-x_2<1;\\ -3x_3, \text{ якщо } x_1-x_2=1;\\ x_1+4x_2, \text{ якщо } x_1-x_2>1 \end{cases}$	21	$y(x_1,x_2,x_3)=\begin{cases} 7x_2-2, \text{ якщо } x_1/x_2<4;\\ 5\sin x_3, \text{ якщо } x_1/x_2=4;\\ 2x_1+3, \text{ якщо } x_1/x_2>4 \end{cases}$
10	$y(x_1,x_2,x_3)=\begin{cases} {x_2}^2, \text{ якщо } x_1<x_3;\\ -(x_1+x_2), \text{ якщо } x_1>x_3;\\ x_1+x_3, \text{ якщо } x_1=x_3 \end{cases}$	22	$y(x_1,x_2,x_3)=\begin{cases} x_1+8, \text{ якщо } x_2/x_3<3;\\ 8x_2, \text{ якщо } x_2/x_3=3;\\ x_3-8, \text{ якщо } x_2/x_3>3 \end{cases}$

11	$y(x_1, x_2, x_3) = \begin{cases} 2x_1 + x_2, \text{ якщо } x_2 = 5; \\ \sqrt{x_1}, \text{ якщо } x_2 < 5; \\ \sin x_1, \text{ якщо } x_2 > 5 \end{cases}$	23	$y(x_1, x_2, x_3) = \begin{cases} x_1 + 2x_2, \text{ якщо } x_3 - x_1 < 3; \\ \sqrt{x_1 \cdot x_2}, \text{ якщо } x_3 - x_1 > 3; \\ 5 / x_1 - x_2, \text{ якщо } x_3 - x_1 = 3 \end{cases}$
12	$y(x_1, x_2, x_3) = \begin{cases} 5x_2 \cdot x_3, \text{ якщо } x_1 > x_2; \\ x_3^2 + 1, \text{ якщо } x_1 = x_2; \\ \cos x_3, \text{ якщо } x_1 < x_2 \end{cases}$	24	$y(x_1, x_2, x_3) = \begin{cases} \lg x_2, \text{ якщо } x_1 / x_3 < 5; \\ \sqrt{x_2 + 2}, \text{ якщо } x_1 / x_3 = 5; \\ x_3 - 9, \text{ якщо } x_1 / x_3 > 5 \end{cases}$

Контрольні питання

1. Яке призначення має мова програмування ПЛК SFC і коли вона застосовується?

2. З яких елементів складається SFC програма? Дайте короткий опис кожного елемента?

3. Що таке розбіжності в програмі SFC і які види розбіжностей бувають?

4. Які умови можуть бути задані в переходах на мові SFC?

5. Що таке дія в мові SFC, і які типи дій бувають?

6. Яке призначення має імпульсна дія в мові SFC, коли застосовуються імпульсні дії?

7. Яке призначення має дія, яка не зберігається в мові SFC, коли застосовуються дії, що не зберігаються?

ЛАБОРАТОРНА РОБОТА № 7

Тема: РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ В СЕРЕДОВИЩІ ISaGRAF

Мета роботи: подальше знайомство з системою технологічного програмування контролерів ISaGRAF і мовою програмування SFC.

Приклад виконання роботи

Постановка задачі. Розробити додаток на мові SFC для віртуального контролера, який реалізує наступний алгоритм управління :

1. При натисканні кнопки "ПУСК" почати процес, при цьому відкрити клапан *A* і завантажити компонент *A* протягом 10 с в ємність *1*.

2. Після закінчення завантаження компонента *A*, при працюючій мішалці *M* почати завантаження компонента, відкривши клапан *B*.

3. Завантаження здійснити протягом 15 с.

4. Після закінчення завантаження компонента *B* суміш продовжити перемішувати ще протягом 10 с.

5. Потім суміш перекачати в ємність *2* включивши насос *H1*. Про спорожнення ємності *1* свідчить спрацювання датчика рівня реле *LS1*.

6. В ємність 2 зробити завантаження компонента *C* протягом 10 с, відкривши клапан С.

7. Дати суміші витриматися протягом 15 с, після чого відкрити клапан *D* і вивантажити готовий продукт із ємності *2*. Про спорожнення ємності *2* свідчить спрацьовування датчика реле рівня *LS2*.

8. Підготувати лінію для виготовлення нової партії продукту.

Рішення задачі

1. Виконати загрузку системи ISaGRAF: Пуск → Програми → ISaGRAF 3.4 → Projects.

2. Створити новий проєкт з ім'ям "pz7": File → New (кнопка Create new project) у вікні Project Management.

3. Відкрити проєкт: 2ЛКМ на імені проєкту pz7 (кнопка Open) вікна Project Management.

4. Створити нову програму: File → New (кнопка Create new program) у вікні Programs.

• Введіть ім'я програми "prog7".

• Виберіть мову "**SFC**: Sequential Function Chart".

• Виберіть стиль "**Sequential:** Main program".

• Натисніть кнопку "Ok".

5. Оголосити використовувані змінні: File → Dictionary (кнопка). Виберіть закладку Integers/Reals, створіть дискретні змінні: a, b, c, d, m, n1 (input, booleans), LS1, LS2 (output, booleans).

6. Виконати редагування тексту програми: 2ЛКМ на імені програми prog7 (кнопка Edit program вікна Programs). Створити програму на мові SFC. Виконати програмування кроків s переходів на мові ST (рисунок 1).

9. Настроїти конфігурацію введення/виведення: Project → I/O connection вікна Programs (кнопка). Виберіть віртуальні плати введення/виведення **xbi8**, **xbo8**.

10. Здійснити прив'язку змінних введення/виведення: Виділіть рядок плати **xbi8** і в меню "Edit" виберіть команду "Set channel/parameter".

11. Створити код програми: Make → Make application вікна Programs (кнопка Make application code). Якщо помилок немає, то натиснути Exit.

12. Виконати симуляцію: Debug → Simulate вікна Programs (кнопка). В меню "Options" вікна симулятора відзначте команду "Variable names".

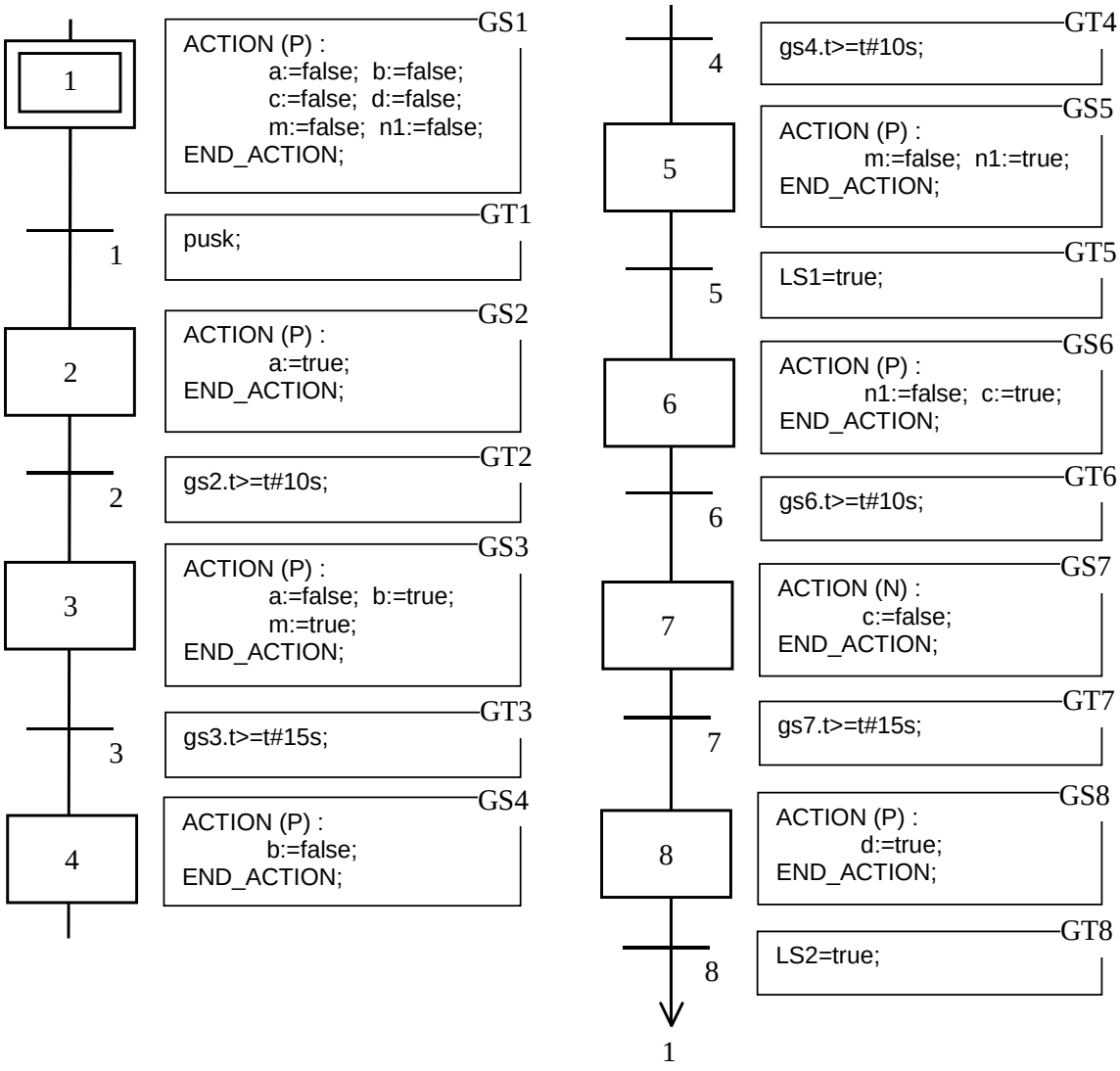


Рисунок 1 – Програма на мові SFC

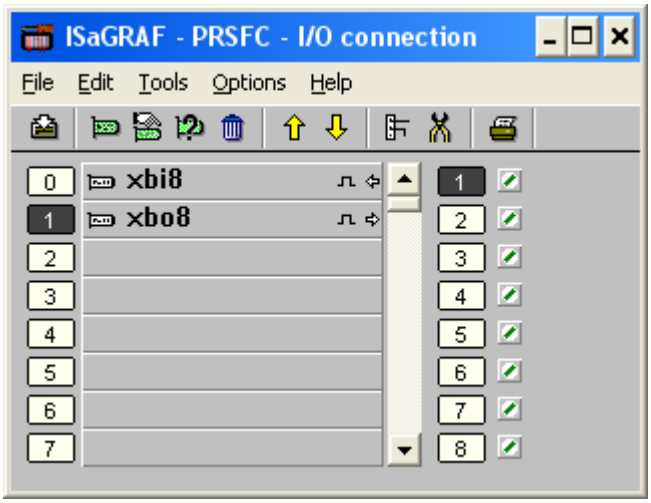
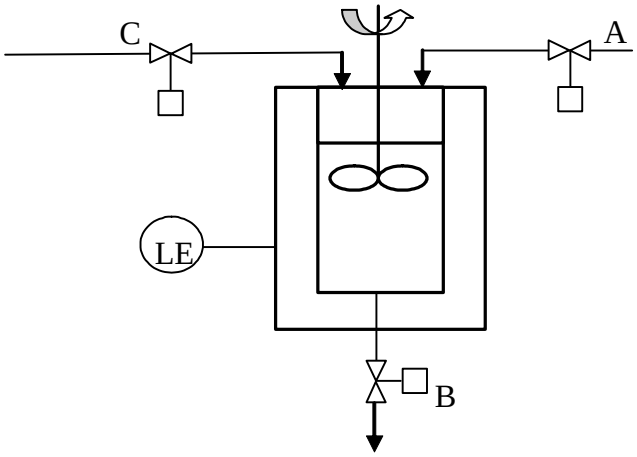


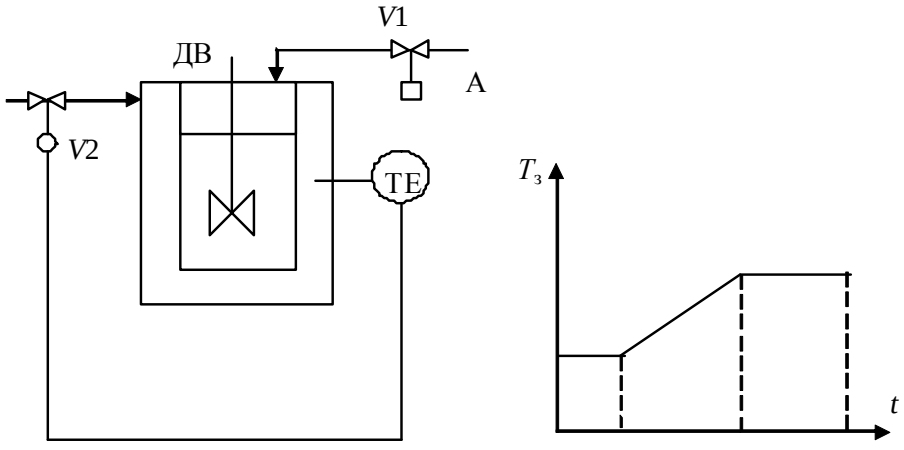
Рисунок 2 – Конфігурація введення/виведення

Варіанти завдань

1. Розробити додаток мовою SFC, що реалізує наступний алгоритм управління. У ємність залити речовину А рівня 1.5 м. Включити мішалку. Потім залити речовину С до рівня 2.6 м. Витримати суміш в апараті протягом 1 години. Вимкнути мішалку і злити отриману суміш В.

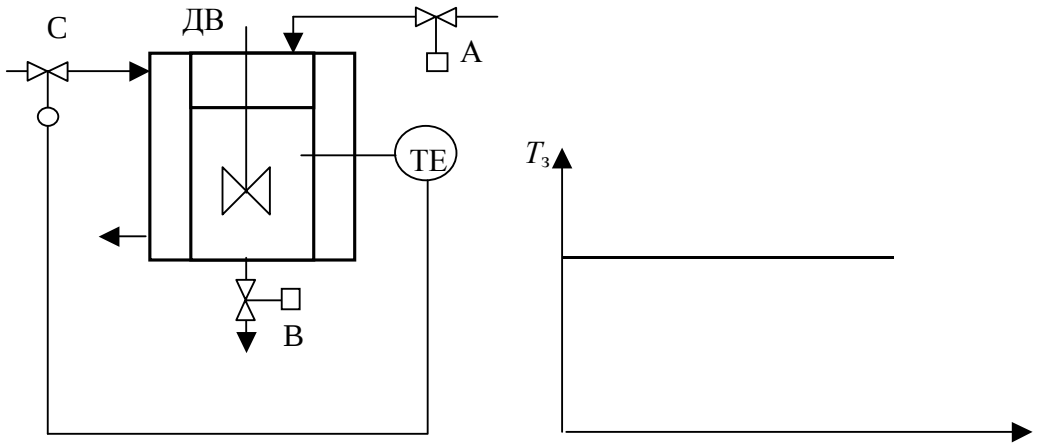


2. Необхідно реалізувати одноконтурну АСР температури у реакторі (а). При цьому необхідно завантажувати речовину А протягом 5 хвилин, потім увімкнути мішалку та підтримувати температуру за заданою програмою (б).



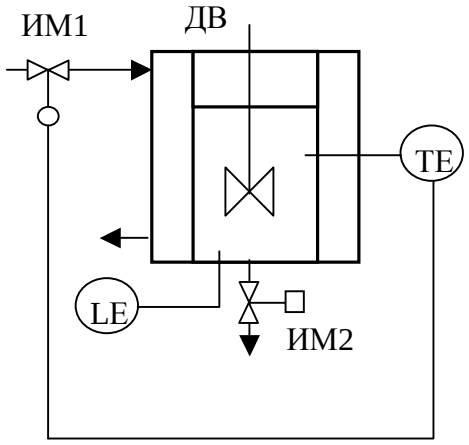
а) б)

3. Необхідно реалізувати одноконтурну АСР температури у реакторі (а). Відкрити клапан А на 30 сек. Потім увімкнути мішалку та підтримувати температуру за заданою програмою протягом 1 години (б). Далі відкрити клапан В.

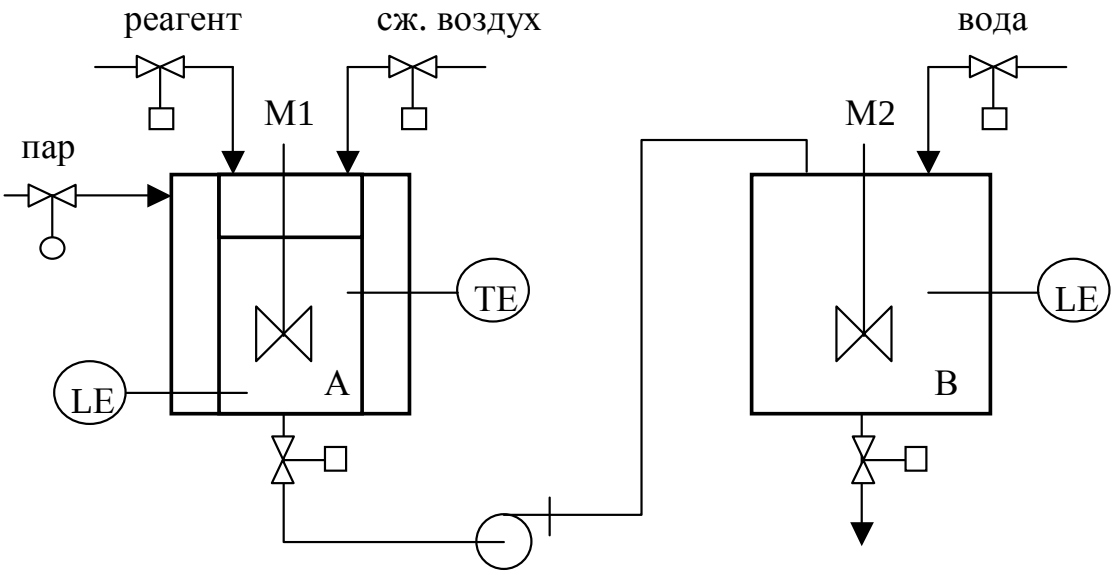


а) б)

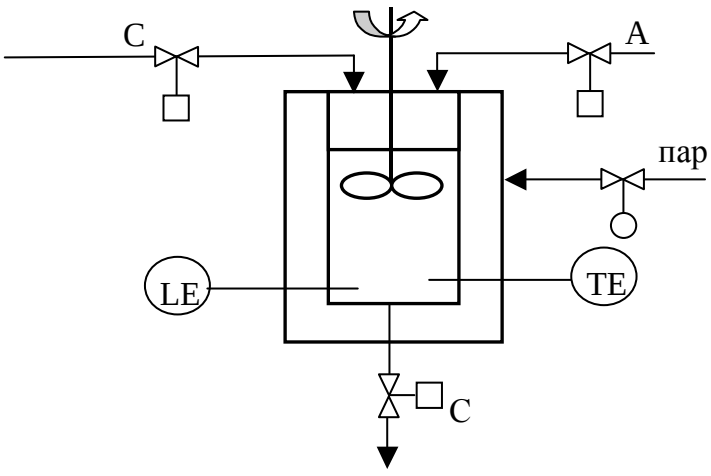
4. Розробити додаток мовою SFC реалізує наступний алгоритм керування машиною, що темперує. Залити компоненти в апарат. Включити мішалку, нагріти до 60 ºС. При цій температурі проводити обробку 4 години. Після цього вимкнути мішалку та перекачати масу.



5. Розробити додаток мовою SFC, що реалізує наступний алгоритм управління. Попередньо апарат А просушити протягом 10 хвилин стисненим повітрям. Потім закачати реагент, увімкнути мішалку, нагріти до температури 80 ºС, вимкнути мішалку, перекачати в апарат В. Одночасно з просушуванням апарату А в апарат В закачати воду і включити мішалку. Перед закачуванням реагенту в апарат злити з воду.



6. Розробити додаток мовою SFC, що реалізує наступний алгоритм управління. У ємність завантажити речовину А рівня 1.5 м. Включити мішалку. Нагріти речовину до 70 ºС. Потім завантажити речовину В до рівня 2.6 м. Витримати суміш в апараті протягом 1 години при температурі 70 ºС. Вимкнути мішалку і злити отриману суміш С.



ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Конюх В.Л. Информационные технологии в автоматизации производства // Вестн. КузГТУ. – 2000.– № 6. – С. 4–9.

2. ISaGRAF: версия 3.4. Руководство пользователя. CJ International, 2003. – 430 с.

3. Деменков Н.П. Языки программирования промышленных контроллеров: Учебное пособие / Под редакцией К.А. Пупкова.- М. : Изд-во МГТУ им.Н.Э.Баумана, 2004.- 172 с.

4. Программируемые контроллеры для систем управления. / Учебное пособие. Часть 1. Архитектура и технология применения. –Х.: ХФИ «Транспорт Украины», 2001. – 316 с.

5. Программируемые контроллеры для систем управления. /Учебное пособие. Часть 2. Характеристики микроконтроллеров и ПЛК. –Х.: ХФИ « Транспорт Украины», 2003. – 264с.

6. Технические средства автоматизации: программирование контроллеров в среде ISaGRAF : лабораторные работы / сост. : И.А. Елизаров, А.А. Третьяков, В.Н. Назаров, М.Н. Солуданов. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2008. – 24 с.

7. http://www.isagraf.ru

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

«ТЕХНОЛОГІЧНЕ ПРОГРАМУВАННЯ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМ УПРАВЛІННЯ»

(для здобувачів вищої освіти денної та заочної форм навчання спеціальності 151 Автоматизація та комп'ютерно–інтегровані технології)

Укладачі:

КУЗНЕЦОВА Олена Володимирівна

ПРОКАЗА Олена Іванівна

Оригінал - макет О.В. Кузнецова

Підписано до друку _____

Формат $60\times84\,{}^{1}\!/_{16}$. Папір типограф. Гарнітура Times.

Друк офсетний. Умовн. друк. арк ._____. Облік.-видавн.арк._____.

Тираж ____ екз. Вид. №_____. Замовл. №_____. Ціна договірна.

Видавництво Східноукраїнського національного університету імені імені Володимира Даля

Адреса видавництва: м. Сєвєродонецьк, просп. Центральний, 59а

Телефон: +38 (050) 218 04 78, факс (064 52) 4 03 42

E-mail:vidavnictvosnu.ua@gmail.com