МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування систем реального часу» для студентів напряму підготовки 6.050202 «Автоматизація та комп'ютерно-інтегровані технології»

Харків
ХНАДУ
2015

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування систем реального часу» для студентів з напряму підготовки 6.050202 «Автоматизація та комп'ютерно-інтегровані технології»

Затверджено методичною радою університету, протокол від . .2015 р.

Харків
ХНАДУ
2015

Укладачі: к.т.н., доц. Філь Н.Ю.

Кафедра автоматизації та комп’ютерно-інтегрованих технологій

ЛАБОРАТОРНА РОБОТА №1
ПОБУДОВА ПРОГРАМИ УПРАВЛІННЯ РОБОТОЮ НАСОСА

1.1. Мета роботи

Розглянути принципи роботи системи розробки ISaGRAF. Ознайомитися з основними принципами побудови, тестування і відладки програми написаної на мові SFC в середовищі ISaGRAF.

1.2. Теоретичні відомості

Мова SFC (Sequential Function Chart) – це графічний мова, яка 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.

У цьому прикладі розглянуті основні операції, необхідні для створення, генерації і тестування короткої тестової програми управління роботою насоса, написаною на мові SCF з ім'ям Simple.

Програма працює по наступному алгоритму: на основі значення двох вхідних булевих змінних в нескінченному циклі активується і деактивується один з кроків процесу і вихідна змінна відображає значення його активності.

1.3. Порядок виконання роботи

Для запуску системи розробки ISaGRAF виконайте команду «Файл→Новий». В результаті відкриється вікно управління проектами.

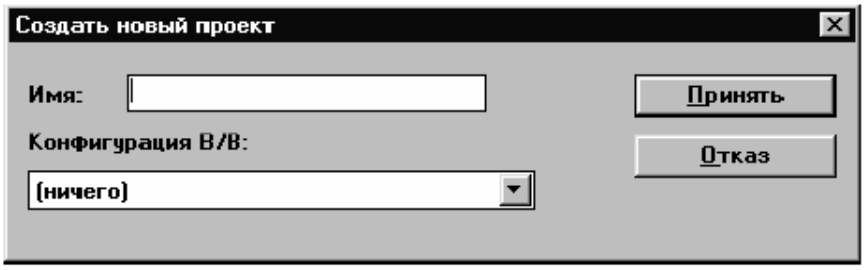


Рисунок 1.1 – Вікно створення нового проекту

Створення проекту.

Створіть проект з іменем «Simple». Для цього в полі «Ім'я» введіть «Simple» і натисніть кнопку «Прийняти». Проект створений і з'явився в списку.

Створення програми.

Двічі клацніть ЛКМ на імені проекту. У діалоговому вікні, що з'явилося, виберіть команду «Файл→Новий». З'явиться вікно «Нова програма». У полі «Ім'я» введіть ім'я програми «Simple», в полі коментарів – «Імітація роботи електронасоса», в полі «Мова» виберіть мову програмування SFC, а в полі «Стиль» – Sequentional (послідовна): Основна програма, після чого натисніть кнопку «Прийняти».

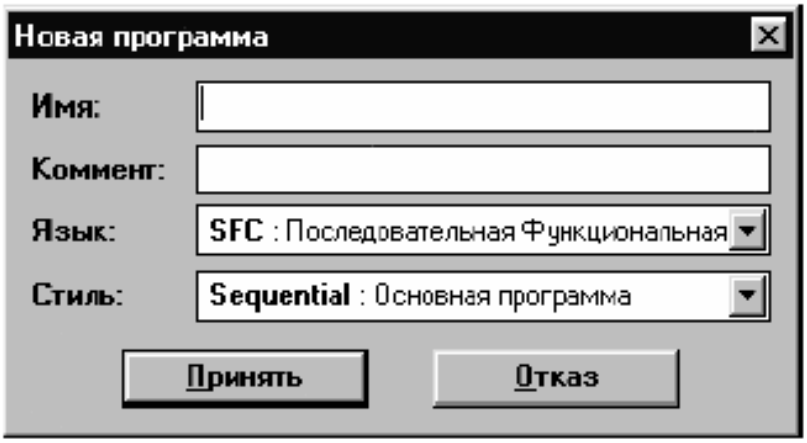


Рисунок 1.2 – Вікно створення нової програми

Програма створена і її ім'я з'явилося у вікні управління програмами.

Оголошення змінних.

Перед введенням тексту програми рекомендується оголосити використовувані в них змінні. Для нашого проекту знадобиться 4 булевих змінні. Їх оголошення виконується за допомогою команд редагування словника. Виберіть команду «Файл→Словник» або двічі клацніть ЛКМ на піктограмі «Словник» панелі інструментів. Відкриється вікно редактора словника «Глобальні Булеві змінні» (рис. 1.3).

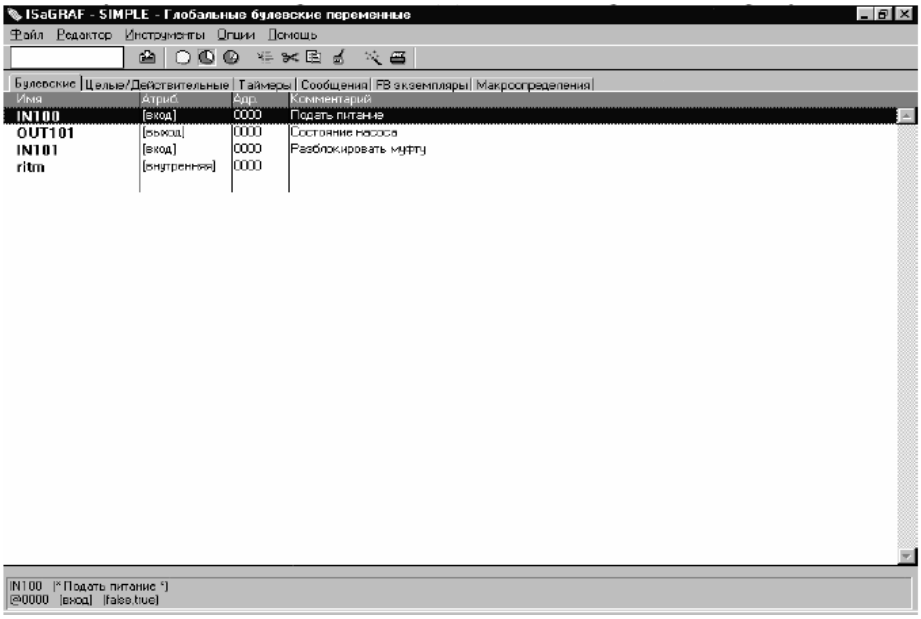


Рисунок 1.3 – Вікно словника

Для створення нових змінних використайте команду «Редактор→Новий» або двічі клацніть ЛКМ на вільному рядку вікна. При цьому з'явиться вікно «Булева змінна», яке використовується для редагування змінних. За описаним нижче способом можна редагувати і інші типи змінних (рис. 1.4).

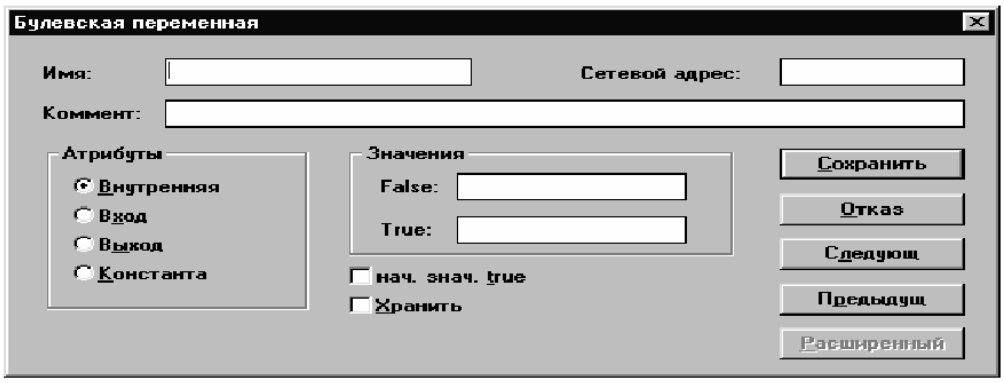


Рисунок 1.4 – Вікно редагування булевих змінних

Здійсніть введення змінних за наступною схемою:

1. Ім'я: IN100, коментарі: «Подати живлення», Атрибути : Вхід. Натисніть кнопку Зберегти.

2. Ім'я: IN101, коментарі: «Розблоковувати муфту», Атрибути : Вхід. Натисніть кнопку Зберегти.

3. Ім'я: OUT101, коментарі: «Стан насоса», Атрибути : Вихід. Натисніть кнопку Зберегти.

4. Ім'я: ritm, Атрибути: Внутрішня. Натисніть кнопку Зберегти.

Для завершення редагування словника виберіть команду «Файл>>Вихід». Збережіть проект командою «Файл>>Зберегти».

Редагування програми.

Двічі клацніть ЛКМ на імені програми Simple. На екрані відкриється вікно SFC-редактора з ім'ям Simple - SFC програма (рис. 1.5).

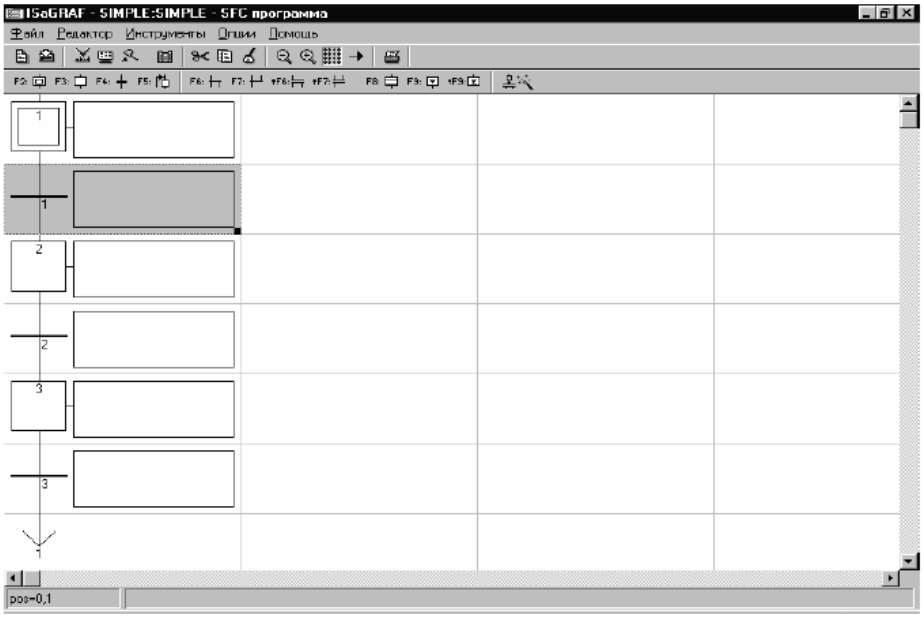


Рисунок 1.5 – Вікно редактора SFC-програми

Налаштуйте вікно, ввівши сітку натисканням кнопки панелі інструментів «Показати/Приховати сітку». У нижній частині вікна зліва показані координати курсору в масштабі ліній сітки, а також початковий крок ініціалізації. Побудуйте частину схеми, як показано на вищенаведеному фрагменті екрану. Використовуйте кнопки панелі інструментів Перехід (F4), Крок (F3), Перехід на крок (F5). При натисканні кнопки «Перехід на крок» з'являється вікно «Призначення стрибка», де вибирається відповідний перехід. Номери кроків і переходів встановлюються автоматично.

Введіть коментарі і текст програми на мові ST у відповідні вікна. Двічі клацніть ЛКМ на комірці pos=0,0 або виберіть команду «Редактор→Редагувати рівень 2», після чого в рядку коментарів з'явилася рамка 1-го кроку GS1, наберіть текст «ініціалізації» (рис. 1.7).

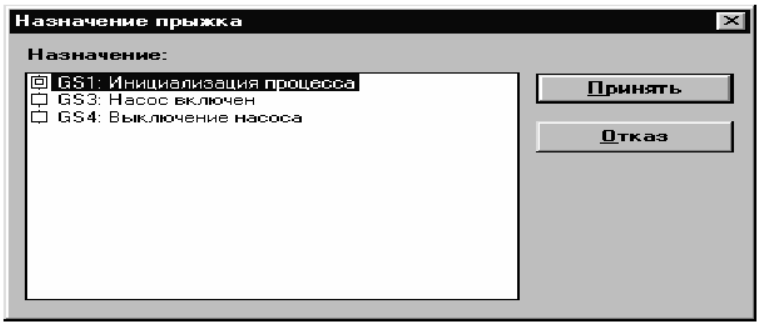


Рисунок 1.6 – Вікно вибору призначення стрибка

Двічі клацніть ЛКМ на 1-му переході GT1 (pos = 0,1), в рядку коментарів наберіть текст «Команда включення насоса», а в рамці редагування тексту програми наберіть текст програми (рис. 1.7).

```
IN100 = true & IN101 = true;
```

Рисунок 1.7 – Фрагмент програми для переходу GT1

Аналогічно для кроку GS2 (позиція pos = 0,2) введіть коментар і текст програми, що для активації вихідної змінної OUT101 при ініціалізації кроку (рис. 1.8).

```
(*насос включен*)

ACTION(P):
OUT101:=true;
END_ACTION;
```

Рисунок 1.8 – Фрагмент програми для блока GS2

Для переходу GT2 (позиція pos=0,3) введіть час виконання кроку 2 (рис. 1.9).

```
GS2.t>t#3s;
```

Рисунок 1.9 – Фрагмент програми для переходу GT2

Для кроку GS3 (pos = 0,4), введіть команду деактивації роботи насоса (рис. 1.10).

```
(*насос выключен*)

ACTION(P):
OUT101:=false;
END_ACTION;
```

Рисунок 1.10 – Фрагмент програми для кроку GS3

Збережіть змінений файл.

Підключення змінних вводу/виводу.

Викличте з вікна управління програмами редактор підключення змінних вводу/виводу і приєднайте їх до пристроїв. Виберіть команду меню «Проект→З'єднання В/В» або клацніть ЛКМ на іконці панелі інструментів «З'єднання В/В». У діалоговому вікні «SIMPLE-З'єднання В/В» виберіть 1-й слот (роз'єм) з номером 12, клацнувши на ньому ЛКМ. Тепер задайте тип плати. Виберіть команду «Редактор→Встановити плату/обладнання» і у вікні «Вибір плат/обладнання» знайдіть плату моделювання булевих входів xbi8: Simulate boolean inputs і клацніть на ній ЛКМ, після чого натисніть кнопку «Прийняти». У вікні SIMPLE-З'єднання В/В на місці установки плати в слоті 12 з'явиться назва плати xbi8, що має 8 булевих входів. Двічі клацніть ЛКМ на 1-м вході плати, після чого з'явиться вікно «З'єднання по каналу В/В №1» зі списком вхідних булевих змінних. Клацніть ЛКМ на змінній IN100 і натисніть кнопку «З'єднати». Потім клацніть ЛКМ на змінній IN101 і натисніть кнопку «З'єднати». Процедура приєднання вхідних змінних до вхідних клем контролера завершена. Натисніть кнопку «Закрити». Опишіть другу плату моделювання булевих виходів. Для цього клацніть ЛКМ на наступному номері посадкового місця (№ 13) і командою «Редактор>>Встановити плату/обладнання» виберіть плату xbo8: Simulate boolean inputs. Потім підключіть вихідну змінну OUT101 до 1-го виходу плати. Вид вікна «SIMPLE-З'єднання В/В» (рис. 1.11).

Закрийте вікно «SIMPLE-З'єднання В/В», зберігши зміни командою «Файл→Зберегти».

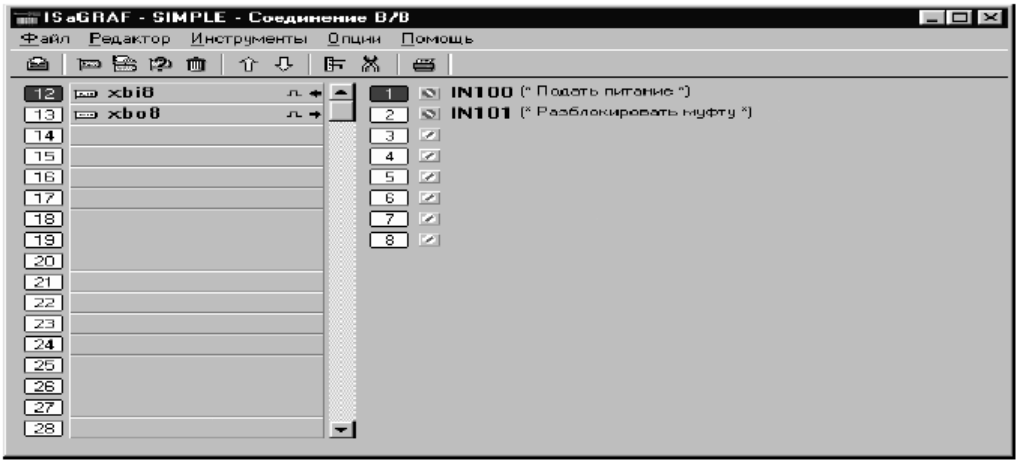


Рисунок 1.11 – Вид вікна приєднання виводів

Побудова коду прикладної програми.

У вікні управління програмами «SIMPLE: SIMPLE-SFC програма» виберіть команду меню «Файл→Перевірити». З'явиться вікно генератора кодів «SIMPLE-Генератор коду», де будуть відображені результати перевірки програми з можливими помилками. Закрийте вікно генератора кодів командою «Файл→Вихід», далі виправте всі знайдені помилки.

Імітація.

У вікні управління програмами «SIMPLE-Програми» виберіть команду меню «Налаштування→Симуляція». З'явиться вікно імітатора програми Simple. Тепер програма може бути протестована. При натисканні обох зелених кнопок IN100, IN101 плати xbi8, на виході плати xbo8 загориться червоний «світлодіод» OUT101, що сигналізує про запуск процесу.

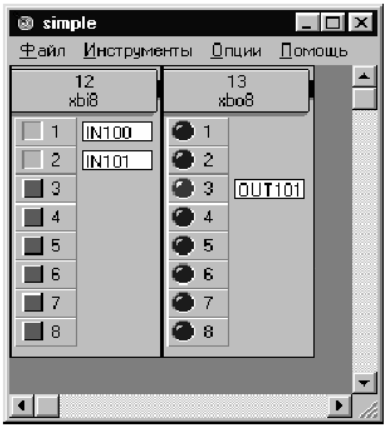


Рисунок 1.8 – Вікно імітації процесу

Для завершення імітації закрийте вікно відладчика командою меню «Файл→Вихід».

1.4. Контрольні запитання

1. Алгоритм оголошення змінних.
2. На якій мові програмування створювалася програма?
3. Яка структура мови програмування SFC?
4. Яка логічна умова дозволяє включення насоса?
5. Як відбувається підключення змінних до мікросхеми?

ЛАБОРАТОРНА РОБОТА №2
ПОБУДОВА ПРОГРАМИ УПРАВЛІННЯ ПРОЦЕСОМ ТЕРМІЧНОЇ ОБРОБКИ

2.1. Мета роботи

Ознайомитися з логікою роботи послідовних і паралельних процедурних кроків роботи програми, а також детальніше розглянути синтаксис мови високого рівня ST.

2.2. Теоретичні відомості

В даному прикладі розглянемо основні операції, необхідні для створення, генерації та тестування проекту управління процесом термічної обробки. Для створення проекту з іменем Attempt будемо використовувати мову програмування SFC, що описує логіку її роботи на рівні послідовних і паралельних процедурних кроків, а також мову високого рівня ST. Основну увагу приділимо синтаксису мов SFC і ST. Проект складається з двох послідовних програм: основної з іменем Main і дочірньої – Temp.

Оброблювана деталь умовно розділена на 4 ділянки, на кожному з яких безперервно вимірюється її середня температура. Значення температури є випадковими величинами. За ціну поділки шкали температури прийнята умовна одиниця. Процес термообробки деталі автоматизований. Для вимірювання температури використовується термопара типу ТХК (термопара хромель-копель), для керування електроприводом і зв'язку з ПК застосовується плата АЦП/ЦАП PCL-818LS фірми Advantech (Тайвань). Якщо температура < 4 од., то до ділянки підводиться

тепло (включається теплоелектронагрівач), якщо >=4 од., то від цієї ділянки тепло відводиться шляхом охолодження у водяній сорочці.

Спрощена схема АСУ ТП термічної обробки показана на рисунку 2.1.

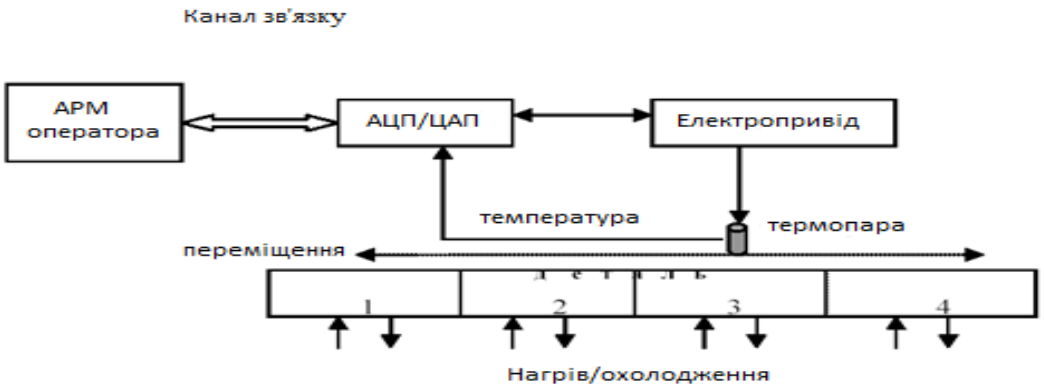


Рисунок 2.1 – АСУ ТП термічної обробки деталі

Технологічний процес передбачає виконання операцій, представлених на схемі алгоритму роботи установки термічної обробки (рис. 2.2).

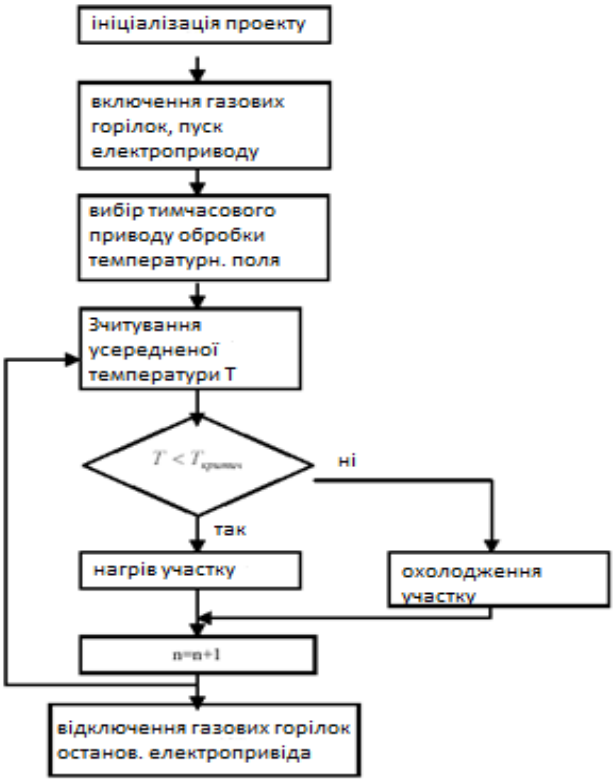


Рисунок 2.2 – Спрощена схема послідовності технологічних операцій при роботі установки термічної обробки

2.3. Порядок виконання роботи

Створіть проект Attempt. Для цього у вікні управління проектами виберіть команду меню «Файл→Новий». У вікні «Створити новий проект» введіть ім'я проекту Attempt і натисніть кнопку «Прийняти».

На 1-му етапі рішення задачі оголосимо змінні по аналогії з пунктом 1.2.4 в попередній роботі. Введемо такі змінні:

Булеві:

Вхідні:

bstart – старт/стоп електроприводу термопари;

gas_on – вкл./викл. газових пальників.

Вихідні:

step1, step2, step3, step4 – фіксовані положення термопари вздовж заготовки;

Heat – включення нагрівача секції;

Cold – включення контуру охолодження секції.

Цілі дійсні:

Вхідні:

timeprog – ціла, значення часу (формат +12, одиниця виміру teath);

randval – ціла, число контурів підігріву/ охолоджування (формат +12).

Вихідні:

nbcycle – ціла (початкове значення 0).

Таймерні:

tmax – внутрішня (t#0s).

Повідомлення:

status – вихідна – основний статус (start/stop).

На 2-му етапі відредагуємо програми на мовах SFC і ST.

Створіть основну і дочірню SFC програми. Для цього виберіть команду меню «Файл→Новий». У діалоговому вікні введіть:

Ім'я: Main

Комент: Основне управління

Мова: SFC

Стиль: Sequential

і натисніть кнопку «Прийняти». Таким чином зареєстрована основна програма Main. Для реєстрації дочірньої SFC програми виконайте ті ж дії з наступними атрибутами діалогового вікна:

Ім'я: Temp

Комент: Процес термообробки

Мова: SFC

Стиль: Дочірня програма: Main

Вид вікна із створеними програмами показаний на рис. 2.4.

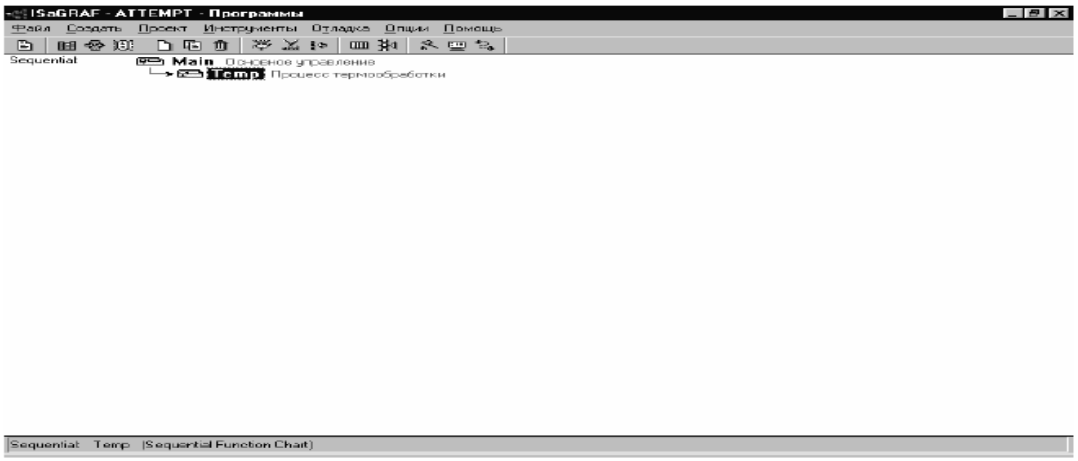


Рисунок 2.3 – Вигляд вікна проекту ATTEMPT

Відредагуйте основну програму Main. Двічі клацніть ЛКМ на імені Main. З'явиться вікно редагування «ATTEMPT: MAIN - SFC програма». У цьому вікні, використовуючи графічний редактор мови SFC, необхідно створити програму (рис. 2.4).

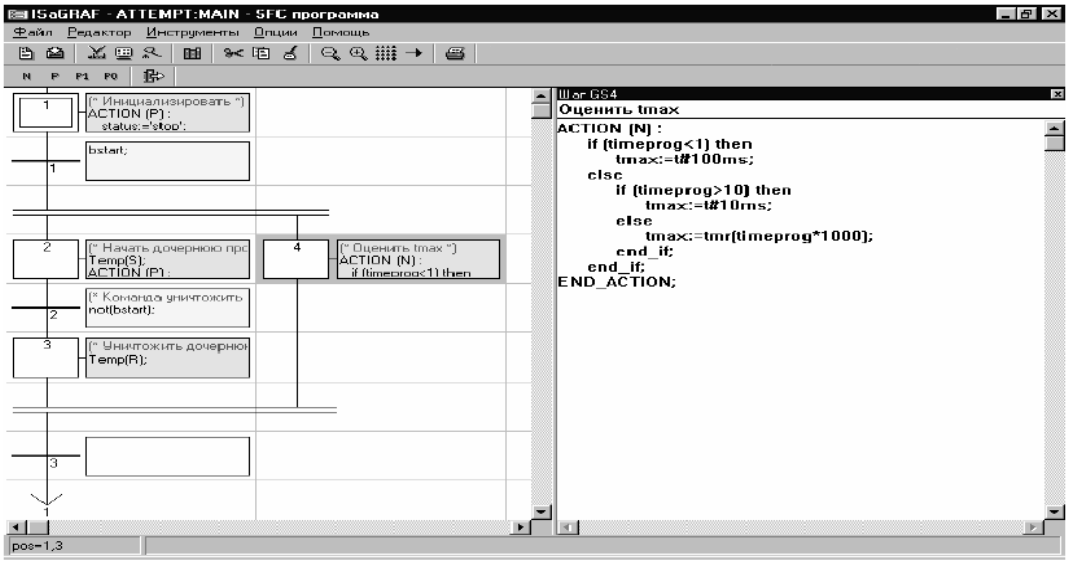


Рисунок 2.4 – Вікно редактора програми Main

Прокоментуємо кроки, переходи, подвійну дивергенцію та конвергенцію.

На 1-му кроці дві вхідні булеві змінні встановлені в false, а змінна – повідомлення status в момент активації кроку приймає значення stop (рис. 2.5).

```
ACTION (P):
status:='stop';
END_ACTION;
```

Рисунок 2.5 – Фрагмент програми для кроку GS1

Умова переходу 1 до наступного кроку – це спільне виконання команд bstart (включення електропривода термопари) та gas_on (подача газу на пальники):

```
bstart = true & gas_on = true;
```

Рисунок 2.6 – Умова переходу до наступного кроку

Далі слідує паралельне виконання операцій програми (подвійна дивергенція). У лівій гілці програми на кроці 2 викликається дочірня програма Temp і змінній status присвоюється значення «start»:

```
ACTION (P):
GSTART(Temp);
status:='start';
END_ACTION;
```

Рисунок 2.7 – Фрагмент програми на другому кроці

Умовою переходу 2 до кроку 3 є переведення вхідних булевих змінних в стан false:

```
bstart = false & gas_on = false;
```

Рисунок 2.8 – Умова переходу до третього кроку програми

На 3-му кроці програми проводиться зупинка дочірньої програми (рис. 2.9)

```
ACTION (P):
GKILL(Temp);
END_ACTION;
```

Рисунок 2.9 – Фрагмент програми зупинки дочірньої програми

На 4-му кроці правої гілки програми для вхідного значення часу timeprog, що встановлюється користувачем, визначається внутрішня змінна tmax – період часу між двома сусідніми положеннями термопари над деталлю:

```
ACTION (N):
   if (timeprog<1) then
         tmax:=t#100ms;
   else
         if (timeprog>10) then
               tmax:=t#10ms;
         else
               tmax:=tmr(timeprog*1000);
         end_if;
   end_if;
END_ACTION;
```

Рисунок 2.10 – Фрагмент четвертого кроку програми

Далі в програмі слідує подвійна конвергенція і перехід на крок 1.

Розглянемо синтаксис дочірньої програми Temp (рис. 2.11).

На 1-му кроці вихідним змінним положення термопари присвоюються значення false, а вихідній змінній кількості ітерацій nbcycle – початкове значення 0 (рис. 2.12).

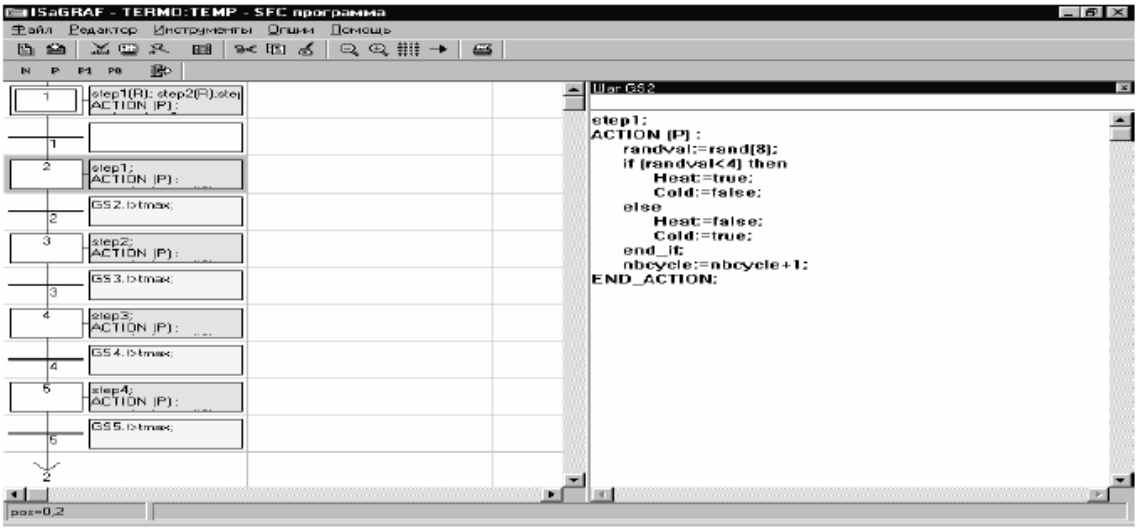


Рисунок 2.11 – Дочірня програма TEMP

```
step1(R); step2(R);step3(R);step4(R);
ACTION (P):
nbcycle:=0;
END_ACTION;
```

Рисунок 2.12 – Фрагмент програми кроку ініціалізації

Після переходу на 2-й крок вихідна змінна step1 встановлюється в стан true, nbcycle отримує одиничне прирощення, проводиться генерація випадкової температури в умовній шкалі 0...7 і за допомогою оператора розгалуження IF-THEN-ELSE включається нагрів або охолодження 1-ї секції деталі (рис. 2.13).

```
step1;
ACTION (P):
   randval:=rand(8);
   if (randval<4) then
         Heat:=true;
         Cold:=false;
   else
         Heat:=false;
         Cold:=true;
   end_if;
   nbcycle:=nbcycle+1;
END_ACTION;
```

Рисунок 2.13 – Фрагмент програми контролю першої секції

Умовою переходу до кроку 3 є затримка на час tmax (рис. 2.13):

```
GS2.t>tmax;
```

Рисунок 2.14 – Умова переходу на наступний крок

На 3-му кроці програми вихідна змінна step2 встановлюється в стан true, генерується випадкова температура в умовній шкалі 0 ... 7 і за допомогою оператора розгалуження IF-THENELSE включається нагрів або охолодження 2-ї секції деталі:

```
step2;
ACTION (P):
   randval:=rand(8);
   if (randval<4) then
         Heat:=true;
         Cold:=false;
   else
         Heat:=false;
         Cold:=true;
   end_if;
END_ACTION;
```

Рисунок 2.15 – Фрагмент програми контролю другої секції

Потім після затримки на час tmax здійснюється перехід до 4-го, а потім до 5-го кроків програми, де вимірюється температура 3-ї і 4-ї секцій деталі.

Нижче показано вигляд вікна з'єднання входів/виходів контролера для 8-ми канальних плат: аналогового входу хаі8 і аналогового виходу хао8.

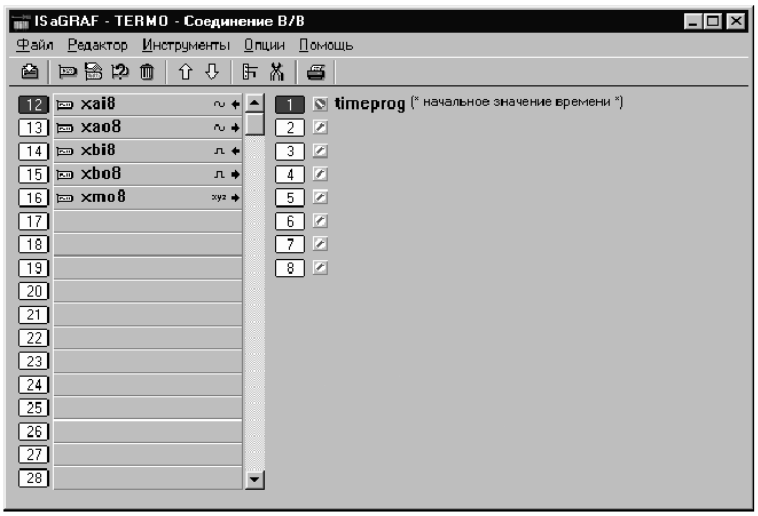


Рисунок 2.16 – Вікно приєднання аналогових входів

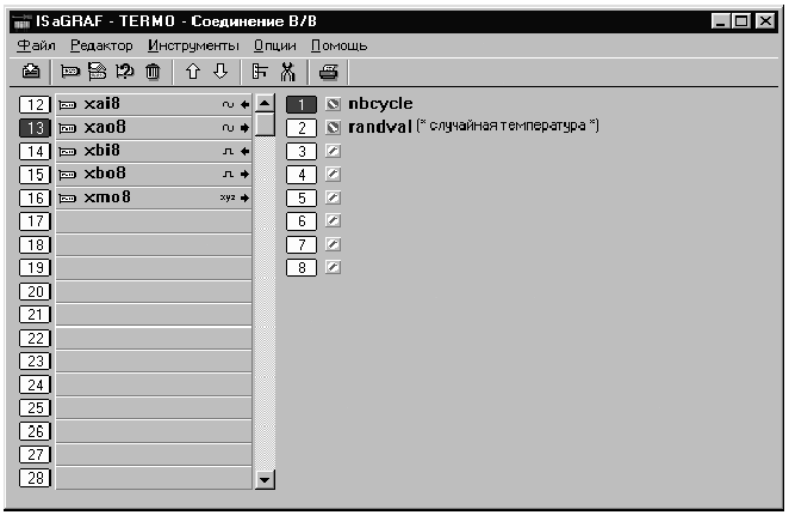


Рисунок 2.17 – Вікно приєднання аналогових виходів

Далі виконується перевірка коду і симуляція програми, вид вікна імітатора програми з відповідними клемами плат вводу/виводу сигналів.

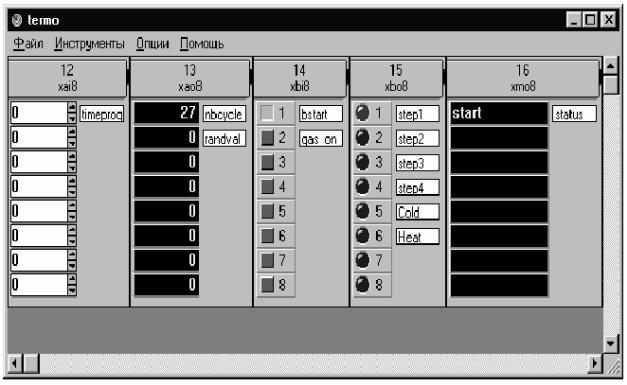


Рисунок 2.17 – Вікно імітації програми

2.3. Контрольні запитання

1. Що у програмі являє собою дивергенція?
2. Що у програмі являє собою конвергенція?
3. В якому фрагменті коду викликається дочірня програма?
4. Вкажіть фрагмент коду, який є умовою початку роботи.
5. Яку функцію виконує дочірня програма Temp?

ЛАБОРАТОРНА РОБОТА №3
ОБРОБКА АНАЛОГОВИХ СИГНАЛІВ ЗА ДОПОМОГОЮ ФУНКЦІЇ ISAGRAF

3.1. Мета роботи

Знайомство з функціями в ISAGRAF. Побудова програми обробки даних з датчика температури.

3.2. Теоретичні відомості

У даному прикладі створюється проект первинної обробки аналогових вимірювальних сигналів, виконуваний за допомогою функції Isagraf. На рисунку 3.1 приведена структурна схема технічних засобів каналу виміру технологічного параметра об'єкту.

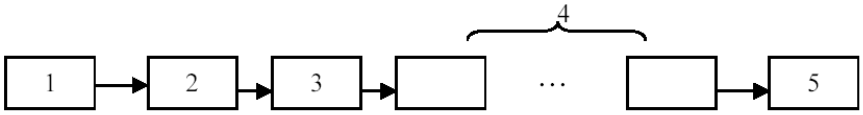


Рисунок 3.1 – Структурна схема технічних засобів каналу виміру технологічного параметра об'єкту

Стан об'єкту 1 сприймається датчиком 2 і перетвориться в стандартний сигнал (наприклад струмовий сигнал 0…20мА) за допомогою вимірювального перетворювача 3. Оскільки не всі датчики мають струмовий вихід, то у вимірювальному каналі встановлюються проміжні перетворювачі 4. Останнім етапом апаратного перетворення є пристрій зв'язку з об'єктом (УЗО) 5, призначений для узгодження виходу вимірювального каналу з входом ЕОМ. Як УЗО може використовуватися аналого-цифровий

перетворювач (АЦП). Для переважної більшості датчиків застосовуються наступні види первинних обробок аналогових сигналів: фільтрація, усереднювання, інтеграція, масштабування, контроль на допустимі значення.

Як приклад розглянемо програмну реалізацію операцій цифрової фільтрації і масштабування аналогового сигналу. Призначення фільтрації полягає в придушенні відносно високочастотних перешкод в лінії зв'язку вимірювального каналу. Широкого поширення набув алгоритм експоненціального згладжування, що задається у вигляді:

$$y_i = \alpha\ x_i - y_{i-1}\ + y_{i-1}, \tag{3.1}$$

де y_i, y_{i-1} – теперішнє та попереднє значення відфільтрованого сигналу, тобто значення на тактах $i, i-1$ роботи АЦП; α – коефіцієнт фільтрації; x_i – теперішнє значення вхідного сигналу.

Цей алгоритм часто представляють у формі:

$$y_i = \alpha \cdot x_i + \ 1-\alpha\ \cdot y_{i-1}. \tag{3.2}$$

Значення y_{i-1} при обробці береться з пам’яті ЕОМ (контролера), нове значення y_i, записується на це місце. Перевага цього методу-простота реалізації, недолік – відносно низька точність.

Масштабування – це вид обробки аналогового сигналу, передбачений для узгодження діапазону вимірювання датчика з діапазоном АЦП, який являється вторинним вимірювальним перетворювачем. Операцію масштабування описує формула виду:

$$y = \frac{A-B}{K} \cdot x + B, \tag{3.3}$$

де y – вимірювана величина в фізичних одиницях; x – вимірювана величина на АЦП; A, B – верхня та нижня границі діапазону вимірювання датчика; K – діапазон коду АЦП.

Рисунок 3.2 пояснює алгоритм обробки аналогового сигналу по формулі (3.3), діапазон виміру датчика нормований і лежить в межах 0…20 мА, АЦП має діапазон в десятковому коді 0…1024,

поточне значення коду вимірюваного сигналу x = 17, що відповідає значенню вимірюваного сигналу в одиницях сили струму y = 0,332 мА.

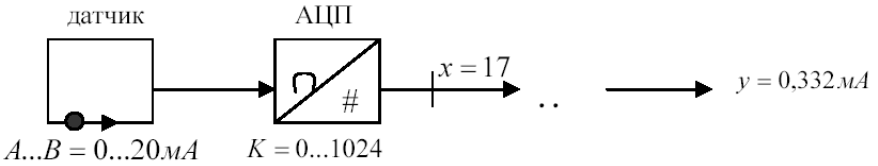
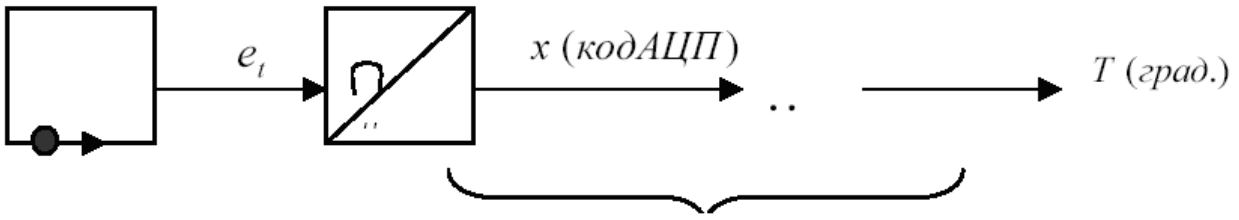


Рисунок 3.2 – Структурна схема обробки сигналу

3.3. Порядок виконання роботи

Розглянемо приклад. Нехай як датчик виміру температури середовища використовується термопара, що має нормуючий перетворювач. Для набуття значень температури при обробці вимірювальної інформації необхідно побудувати таблицю перетворення на основі характеристики даного типу термопари $e_t = F(T)$, де e_t – термо-ЕДС; T – температура середовища. Даний алгоритм пояснюється наступним рисунком.



Таблиця перетворень АЦП

Рисунок 3.3 – Послідовність перетворень коду

Створимо проект з іменем Function, що реалізовує операції масштабування і цифрової фільтрації вимірювального сигналу. Проект складатиметься з трьох програм: програми Step на мові ST з атрибутом Begin, програми Main на мові SFC з атрибутом Sequential і підпрограми з ім'ям adc2val з атрибутом Function. Функція задає параметри підпрограми і алгоритм обробки вимірювального сигналу, програма Step визначає значення змінних підпрограми-функції. При виконанні дій в тілі основної програми Main використовуються значення поверненої змінної підпрограми-

функції. Після створення проекту Function, вікно програм після використання команди «Файл→Новий» має наступний вигляд.

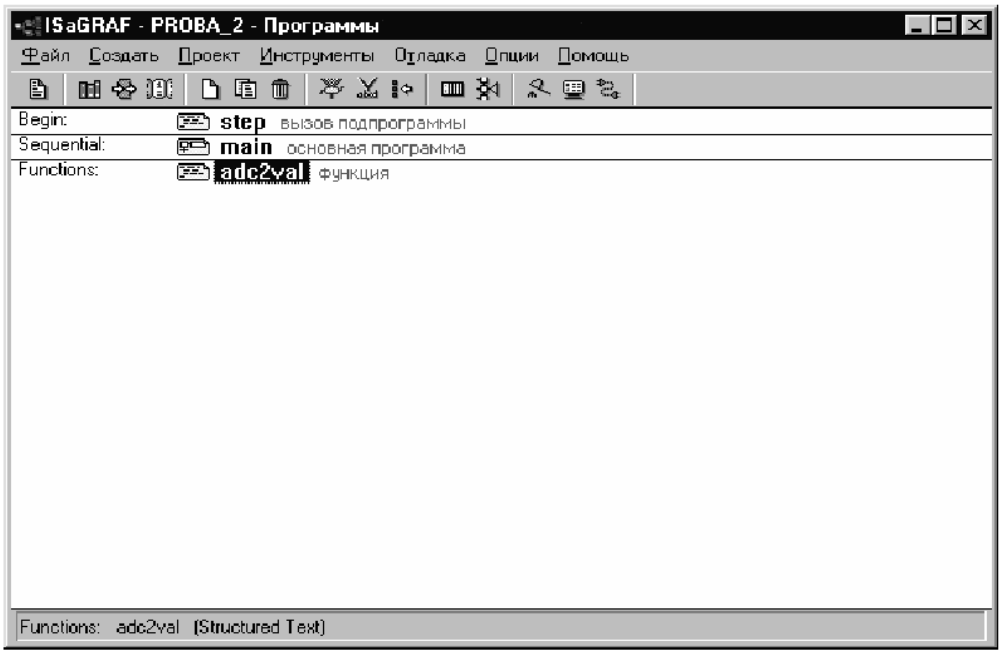


Рисунок 3.4 – Вікно проекту Function після створення програм

При створенні функції необхідно задати коментарі і атрибути відповідно до рисунку.

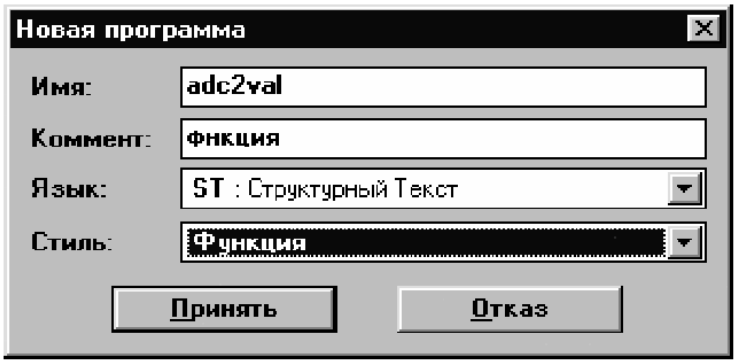


Рисунок 3.5 – Створення функції

У програмах використовуватимуться глобальні і локальні змінні.

У словнику змінних визначимо глобальні змінні:

вхідні: **bo1** – булева; **in1** – ціла;

вихідні: **out** – дійсна; внутрішні: **p_in** – дійсна.

Для опису параметрів підпрограми використовується команда «Файл>Параметри». Параметрами функції, що викликаються, є:

code_adc (аналоговий), **min_adc** (аналоговий), **max_adc** (аналоговий), **min_sign** (дійсний), **max_sign** (дійсний), **kf** (дійсний), **o_val** (дійсний). Значенням що вертає функція (для функції лише одне) є однойменна з іменем функції змінна adc2val (дійсна). Вікно параметрів функції (рис. 3.6).

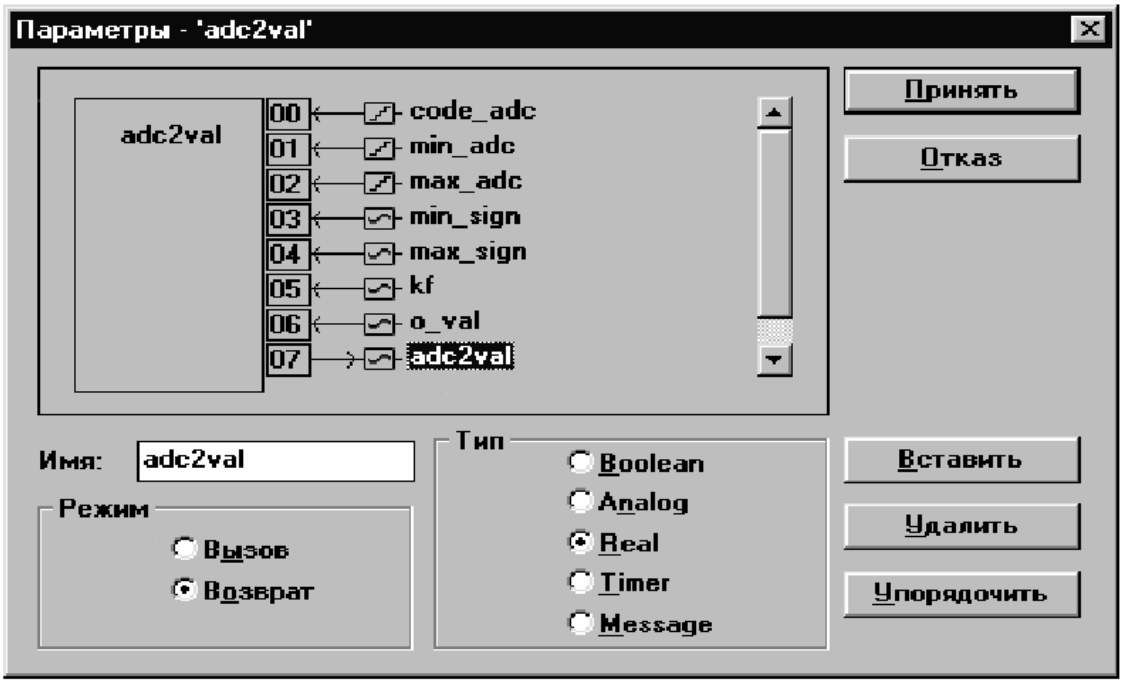


Рисунок 3.6 – Вікно параметрів функції

Двічі клацнувши ЛКМ на імені функції у вікні програм, здійснюється перехід в тіло функції для опису обробки аналогового сигналу відповідно до формул (3.1) – (3.3) на мові ST (рис. 3.7)

```
adc2val:=(real(code_adc)*(max_sign-min_sign)/real(max_adc-
min_adc)+min_sign)*(1.0-kf)+o_val*kf;
```

Рисунок 3.7 – Код обробки аналогового сигналу

Перевірка програми adc2val здійснюється по команді «Файл>Перевірити» у вікна редагування програм. Програма Step звертається до підпрограми adc2val, визначаючи значення змінних, при чому змінним присвоюються значення з нижченаведеної таблиці відповідності за принципом: поточний параметр функції, що викликається – значення аргументу підпрограми, що викликається.

Таблиця 3.1 – **Відповідність параметрів функції**

Параметр, що викликається	Значення параметра
code_adc	in1
min_adc	0
max_adc	65535
min_sign	0.0
max_sign	10000.0
Kf	0.5
o_val	p_in

Програма з іменем Step складається з оператора виклику підпрограми – функції (рис. 3.9).

```
p_in:=adc2val(in1,0,65535,0.0,10000.0,0.5,p_in);
```

Рисунок 3.9 – Текст програми Step

У основній програмі Main при ініціалізації вхідної булевої змінної bo1, вихідній змінній out присвоюється значення відфільтрованого і промасштабованого значення вимірюваної величини (рис. 3.10).

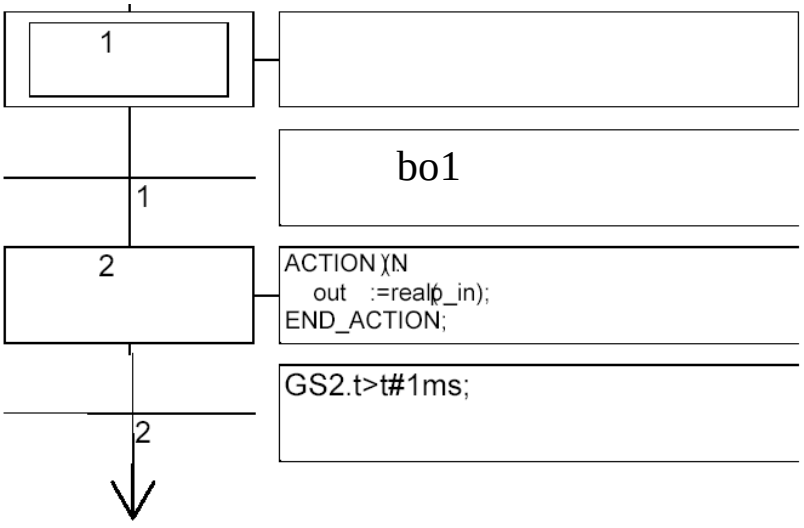


Рисунок 3.10 – Структура програми Main

Приєднання змінних до плат вводу/виводу здійснюється по аналогії з попередніми роботами. Якщо всі дії при створенні

проекту виконані коректно, з урахуванням приведених рекомендацій, то після компіляції проекту буде видано повідомлення (рис. 3.11).

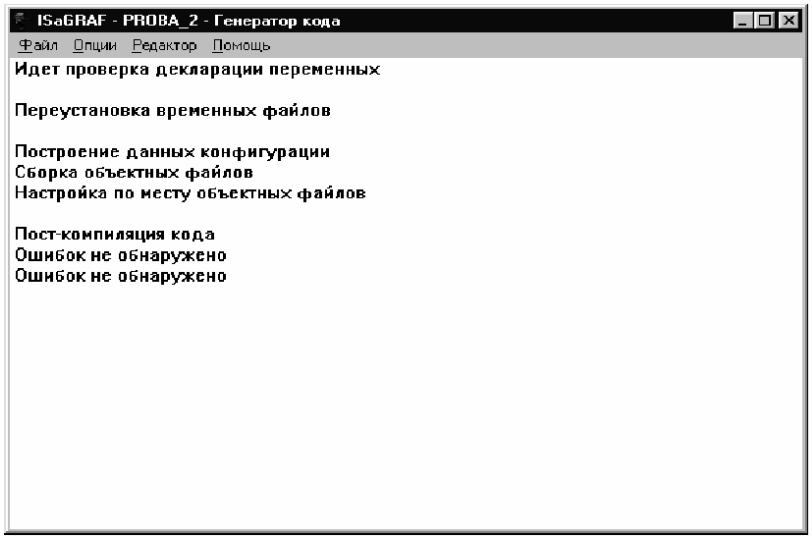


Рисунок 3.11 – Вікно успішної компіляції

Відладка програми виконується по команді «Відладка→Симуляція».

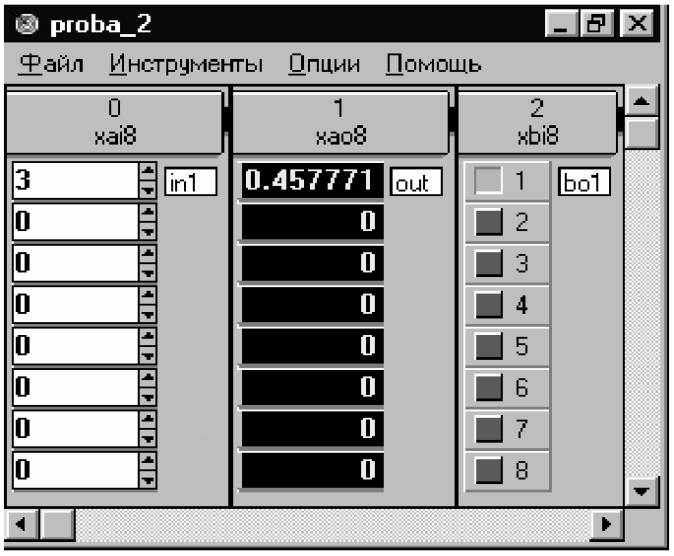


Рисунок 3.12 – Вікно симуляції програми

3.4. Контрольні запитання

1. Опишіть послідовність перетворень при вимірюванні.
2. Що являє собою програма adc2val?
3. Звідки беруться значення параметрів для функції adc2val?

4. Яке призначення кожної з трьох програм проекту?
5. Чим відрізняються локальні та глобальні змінні?

ЛАБОРАТОРНА РОБОТА №4
ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНИХ БЛОКІВ

4.1. Мета роботи

Ознайомитись з роботою функціональних блоків ISaGRAF

4.2. Теоретичні відомості

В даному прикладі створюється проект, який використовує підпрограму у вигляді функціонального блока. Функціональні блоки можуть використовувати наступні мови: LD, FBD, ST або IL. Середовище ISaGRAF має функціональні блоки стандартної бібліотеки. Функціональні блоки може визначати й користувач. У відмінності від функції, функціональний блок може мати декілька змінних, що повертаються. Локальні змінні функціональних блоків у словнику не оголошуються. У словнику оголошується ім’я екземпляра функціонального блока.

4.3. Порядок виконання роботи

Створимо проект з іменем f_block та складається з трьох програм: start (має атрибути Begin, ST), main (має атрибути End, ST) і функціональний блок fbb_1 (має атрибут Функціональний блок, ST) (рис. 4.1).

Командою «Файл→Параметри» у вікні програм визначимо конфігурацію функціонального блока fbb_1. Локальними змінними що викликаються є: x1, x2 (дійсні), змінними що повертаються – Q1, Q2 (дійсні) (рис. 4.2).

Дії у вікні функціонального блока аналогічні операціям конфігурування функції у роботі 3. Двічі клацніть ЛКМ на ім’я функціонального блока або за командою «Файл→Відкрити» у вікні редагування функціонального блока та визначить операції над вхідними змінними за допомогою операторів присвоєння (рис. 4.3).

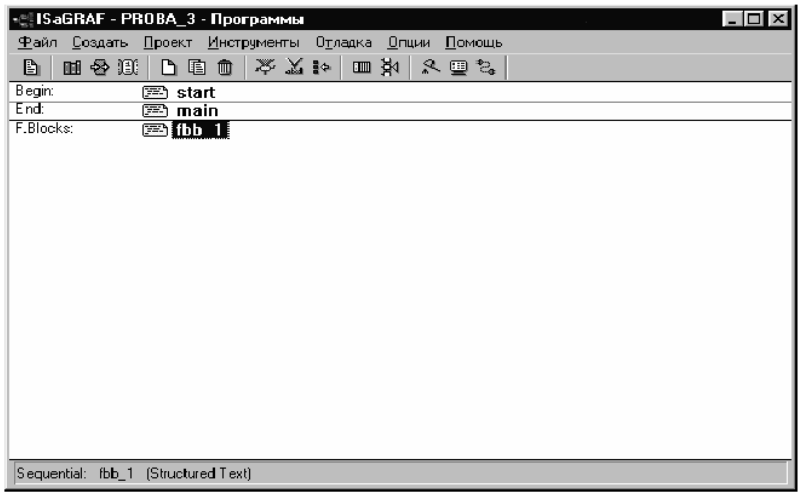


Рисунок 4.1 – Вікно проекту f_block

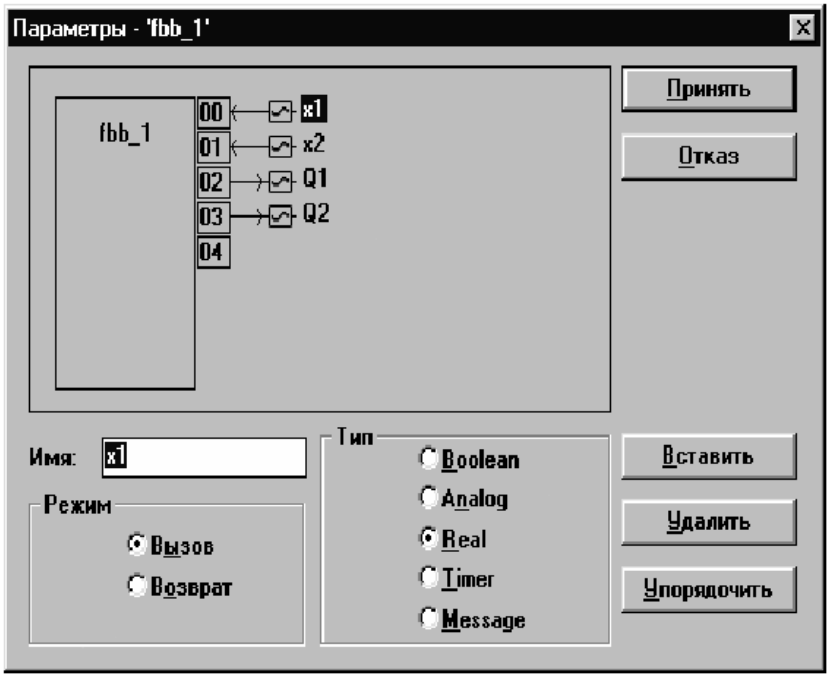


Рисунок 4.2 – Вікно конфігурації функціонального блоку

```
Q1:=x1*x2+sin(x1+x2);
Q2:=sqrt(x1)-log(x2);
```

Рисунок 4.3 – Визначення операцій над вхідними змінними

У програмі Start оголошується екземпляр функціонального блока. Екземпляр може мати будь яке відмінне від fbb_1 ім’я. Синтаксис оператора оголошення наступний:

<ім’я екземпляра>(<змінна, що викликається 1>,<змінна, що викликається 2>,…,<змінна, що викликається n>);

У нашому випадку ім’я екземпляра fbbu.

Оператор виклику значення, що повертається , функціонального блока має вигляд:

<ім’я екземпляра>.<значення, що повертається>;

Програма Start складається з трьох операторів (рис. 4.4).

```
fbbu(T1,T2);
A1:=fbbu.Q1;
A2:=fbbu.Q2;
```

Рисунок 4.4 – Оператори програми Start

У програмі Main вихідним змінним out1 і out2 привласнюються значення функціонального блока (рис. 4.5).

```
out1:=A1;
out2:=A2;
```

Рисунок 4.5 – Фрагмент коду програми Main

За аналогією з вищерозглянутими прикладами здійснюється оголошення у словнику глобальних змінних й екземпляра функціонального блока:

− внутрішні змінні: **A1, A2** (дійсні);

− вхідні змінні: **T1, T2** (дійсні);

− вихідні змінні: **out1, out2** (дійсні).

Для оголошення екземпляра функціонального блока у словнику використовується закладка «FB екземпляри» (рис. 4.6).

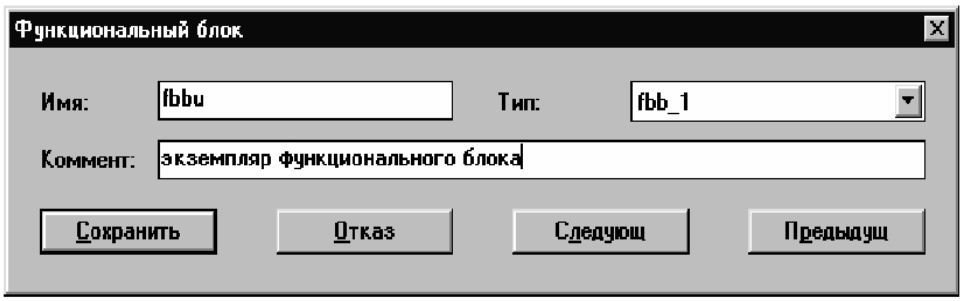


Рисунок 4.6 – Оголошення функціонального блока

Після перевірки програми та її компілювання розглянемо її роботу в режимі імітації (рис. 4.7).

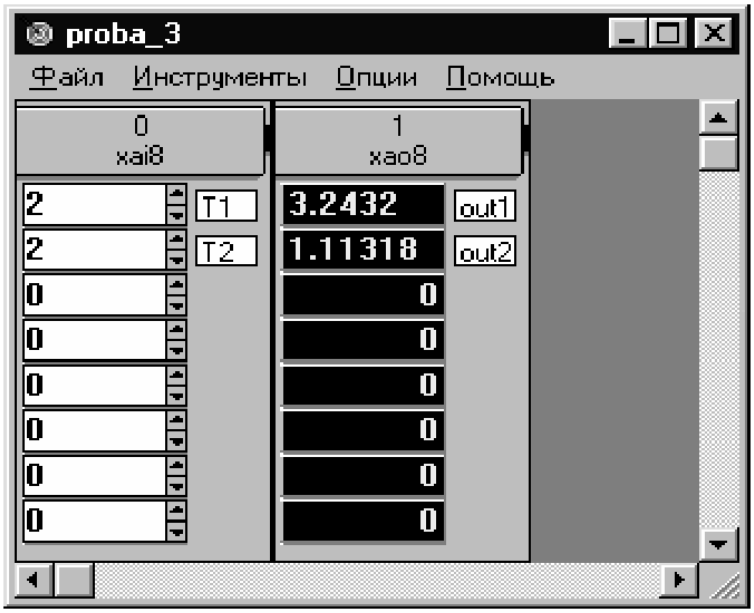


Рисунок 4.7 – Вікно імітації програми

4.4. Контрольні питання

1. Що являє собою функціональний блок?
2. Чим відрізняються функція і функціональний блок?
3. Які мови програмування можуть бути використані у FB?
4. Як проголошується екземпляр функціонального блока?
5. Чи використовуються нестандартні функціональні блоки?

СПИСОК ЛІТЕРАТУРИ

1. Колтунцев А.В. Стандарт IEC 61499 и система программирования контроллеров ISaGRAF 5: от теории к практике [Текст] / А.В. Колтунцев, С.В. Золотарев // Rational Enterprise Management. − 2009. − №2. − С. 24-31.

2. Яковлев А.В. Расширения ISaGRAF 5: инновационные функциональные возможности, производительность и открытость [текст] / А.В. Яковлев, А.В. Липовец, С.В. Золотарев // ИСУП. − 2009. − №2. С. 11-15.

3. Золотарев С.В. Некоторые особенности реализации стандарта IEC-60870-5-104 в системе программирования контроллеров ISAGRAF: от теории к практике [текст] / С.В. Золотарев // ИСУП. − 2010. − №4. С. 7-14.

4. Карпов Ю.Г. Теория автоматов [текст]/ Ю.Г. Карпов. − СПб.: Питер, 2002. − 224 с.

ЗМІСТ

Навчальне видання

Філь Н.Ю.

Методичні вказівки
до виконання лабораторних робіт з дисципліни
«Програмування систем реального часу»
для студентів напряму підготовки 6.050202
«Автоматизація та комп'ютерно-інтегровані технології»

Відповідальний за випуск Нефьодов Л.І.
Редактор

План 200_, поз._____
Підписано до друку_____ Формат 60х84 1/16.
Умовн. друк арк._____ Обл. – вид. Арк. _____
Замовлення №_____ Тираж ___прим. Ціна договірна

ХНАДУ, 61002, Харків, вул. Петровського, 25

Свідоцтво державного комітету інформаційної політики, телебачення та радіомовлення України про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції,
серія № ДК № 407

Підготовлено і надруковано видавництвом
Харківського національного автомобільно-дорожнього університету