МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька Політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт за темою «Основи програмування завдань керування об'єктом у середовищі програми OpenPLC» з дисципліни «Системи керування та контролю електромеханічних пристроїв та систем» для студентів усіх форм навчання спеціальності 141 Електроенергетика, електротехніка та електромеханіка, освітні програми – Електричні та електронні апарати, Електромеханічне обладнання енергоємних виробництв

2024

Методичні вказівки до виконання лабораторних робіт за темою «Основи програмування завдань керування об’єктом у середовищі програми OpenPLC» з дисципліни «Системи керування та контролю електромеханічних пристроїв та систем» для студентів всіх форм навчання спеціальності 141 Електроенергетика, електротехніка та електромеханіка, освітні програми – Електричні та електронні апарати, Електромеханічне обладнання енергоємних виробництв /Укл.: М.О. Поляков, – Запоріжжя: НУ «Запорізька політехніка», 2024. – 22 с.

Укладач: М. О. Поляков, професор, д.т.н.

Рецензент: В. В. Василевський, доцент, к.т.н.

Відповідальний
за випуск: П. Д. Андрієнко, професор, д.т.н.

Затверджено
на засіданні кафедри
«Електричні та електронні
апарати»
Протокол № 7
від 26.12.2023 р.

Затверджено
НМК ЕТФ
Протокол №7
від 21.03. 2024 р.

ЗМІСТ

ВВЕДЕННЯ

Метою виконання лабораторних робіт є закріплення на практиці теоретичного матеріалу з дисципліни «Системи керування та контролю електромеханічних пристроїв та систем». Для розробки додатків керування використовуються пакети програм OpenPLC Editor, OpenPLC Runtime.

Основна проблема в галузі інженерного навчання програмування промислових контролерів полягає в недоступності з економічних міркувань, як самих контролерів, так і середовищ для програмування мовами стандарту МЕК 61131-3.

У той самий час щодо алгоритмів управління об'єктами широко застосовуються контролерні плати сімейств Arduino та інших, які доступні для придбання. При цьому використовується мова програмування С та не використовуються мови стандарту МЕК 61131-3, що є недоліком такого підходу.

У цих методичних матеріалах будемо використовувати вільно розповсюджувану платформу OpenPLC для навчання програмуванню мовами стандарту МЕК 61131-3 з виконанням розроблених програм, як у вбудованому логічному контролері, так і реальних контролерних платах, наприклад із сімейства Arduino.

Платформа OpenPLC включає середовище програмування універсальною (не прив'язаною до конкретної платформи виробника контролерних плат) мовою та засіб завантаження програми в реальну плату OpenPLC RunTime.

При конфігуруванні проекту в середовищі OpenPLC Editor здійснюється вибір мови програмування. Можливі варіанти мов: мова інструкцій (IL), структурованого тексту (ST), сходових діаграм (LD), функціональних блокових діаграм (FBD) та послідовних функціональних кроків (SFC).

Методичні вказівки містять опис лабораторних робіт.

Студент зобов’язаний вивчити теоретичний матеріал, виконати практичні завдання та зробити звіт з лабораторної роботи, який буде мати основні розділи: - назва лабораторної

роботи; її мета; схеми експериментів, опис додатків, тексти програм таблиці, графіки, розрахунки, висновки.

Звіт має бути оформлений у відповідності з ДСТУ 3008 - 2015 і захищений.

Лабораторна робота 1
Мова LD. Базові інструкції мови програмування LD. Робота з емулятором контролера та інструкції обробки даних

Мета. Вивчити елементи мови програмування LD, інструкції контролера бітові, таймерів, лічильників, порівняння обробки даних (арифметичні, тригонометричні, логічні, перетворення тощо).

Короткі теоретичні дані. Мова LD – це одна з п'яти стандартних мов програмування контролерів за стандартом IEC61131-3. LD програма складається з рангів. Ранг має умовну та виконавчу частини. Частини рангу складаються з гілок. Гілка, у свою чергу, має вхід, вихід та містить інструкцію контролера – умовну чи виконавчу. Дія умовної інструкції це перевірка істинності певної умови, а вихідної – у зміні значення програмних тегів. Кожна інструкція виконує певну функцію за якої її можна віднести до категорії базових інструкцій, інструкцій обробки даних тощо. Загальна кількість типів інструкцій залежить від типу контролеру.

Програма мовою LD візуально нагадує релейно-контактну схему електроавтоматики. Приклад простої програми для реалізації функції реле із самоблокуванням наведено на рис. 1.1.

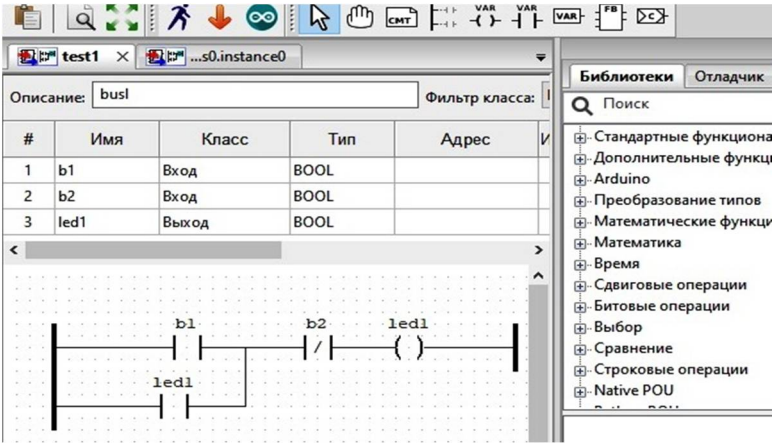


Рисунок 1.1 – Програма для реалізації функції реле із самоблокуванням

За допомогою інструкцій таймерів TON, TOF та лічильників CTU, CTD виконують формування часових інтервалів та підрахунок кількості подій. Приклад програми, що формує імпульси та підраховує їх кількість наведено на рис. 1.2.

Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	Опис
blink_led	Локальный	BOOL				
reset	Локальный	BOOL				
TON0	Локальный	TON				
TOF0	Локальный	TOF				
CTU_DINT0	Локальный	CTU_DINT				
CV_count	Локальный	DINT				

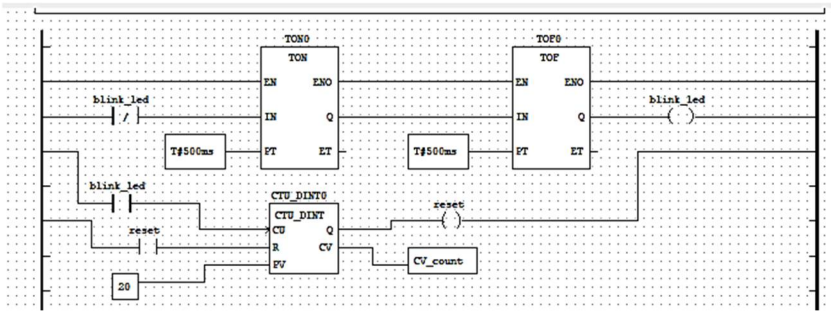


Рисунок 1.2 – Програма що формує імпульси та підраховує їх кількість, та скидає лічильник за певної умови

За допомогою інструкцій ADD, SUB, DIV, MUL виконують арифметичні дії. Тригонометричні інструкції COS, TAN, SIN, ATN, ASN та інші використовують для обчислення прямих та зворотних тригонометричних функцій. Логічні інструкції AND, OR, XOR, NOT виконують логічні дії над бітами операндів. Група інструкцій перетворення виконують перетворення типу «радіан – градус», «двійковий – двійкове десятинний код», «двійковий - унітарний код». Короткий опис інструкцій наведено у вікні «Библиотеки».

Хід роботи

1. У середовищі програмування OpenPLC створити проект на мові LD.

2. Розробити програму, що реалізує логічну функцію $y = \overline{(x_1 \wedge x_2)} \vee x_3$, де x_1, x_2, x_3 – цифрові(булеві) вхідні змінні, що надходять до контролера через його входи; y – цифрова вихідна змінна, значення якої формується контролером. Виконати програму у симуляторі для усіх комбінацій вхідних змінних. За результатами виконання скласти таблицю істинності логічної функції. Розробити блок-схему алгоритму програми, що реалізує логічну функцію.

3. Виконати попередній пункт щодо функції, яка задана викладачем.

4. Розробити та завантажити в емулятор контролеру програму що формує використовуючи інструкцію TON інтервали часу певної тривалості, підраховує кількість цих інтервалів за допомогою лічильнику CTU, та виконує скидання лічильнику у моменти часу, коли акумулятор лічильнику досягає певного значення. У звіті навести програму та графіки зміни основних змінних програми від часу.

5. Розробити та виконати за допомогою емулятору програму, що реалізує функції реверсивного лічильнику. Напрям зміни станів лічильнику задавати за допомогою вхідної змінної логічного типу. Виконати автоматичну зміну значення цієї змінної таким чином щоб значення акумулятору лічильнику не виходило за межі інтервалу [$N_{\min}$, $N_{\max}$]. У звіті навести програму та графіки зміни основних змінних програми від часу.

6. Модернізувати попередню програму додавши до неї керовані вихідні змінні логічного типу, які набувають значення

«1» якщо виконуються умови щодо значення акумулятору лічильнику N:

- $N > N_1$, де $N_1 \in [N_{\min}, N_{\max}]$;
- $N < N_1$, де $N_1 \in [N_{\min}, N_{\max}]$;
- $N_1 < N < N_2$, де $N_1 \in [N_{\min}, N_{\max}]$, $N_2 \in [N_{\min}, N_{\max}]$.

Виконання умов перевіряти за допомогою інструкцій порівняння. У звіті навести програму та графіки зміни основних змінних програми від часу.

7. Розробити та завантажити в емулятор контролеру програму, що за двома катетами обчислює гіпотенузу трикутника.

8. Розробити та завантажити в емулятор контролеру програму, що формує імпульси шляхом зміни на протилежний певних біт слова за допомогою інструкції XOR.

Контрольні питання

1. Склад, призначення та параметри бітових інструкцій.
2. Склад, призначення та параметри інструкцій таймерів.
3. Склад, призначення та параметри інструкцій лічильників.
4. Склад, призначення та параметри інструкцій порівняння.
5. Склад, призначення та параметри арифметичних інструкцій.
6. Склад, призначення та параметри логічних інструкцій.
7. Склад, призначення та параметри тригонометричних інструкцій.
8. Склад, призначення та параметри інструкцій перетворення.
9. Програмування булевої функції.

Лабораторна робота 2
Програмування завдань керування мовами стандарту МЕК 61131-3

Мета. Вивчити елементи мов програмування LD, FBD, SFC, ST, IL та методи побудови функцій та функціональних блоків у програмі керування об'єктом що використовує декілька мов.

Короткі теоретичні дані. Мови програмування LD, FBD, SFC, ST, IL за стандартом МЕК 61131-3 використовуються для програмування промислових контролерів. Ці мови з різною ефективністю вирішують завдання керування об'єктом. Тому у додатку керування може використовуватись декілька мов одночасно. На рис. 2.1 наведено проект у якому основна програма *plc_prog* написана мовою FBD, функціональні блоки - мовами LD, FBD, SFC, ST, IL, а функція *AverageVal* - мовою ST. На рис.2.2 наведено структуру програми plc_prog яка обчислює середньоарифметичне значення лічильників.

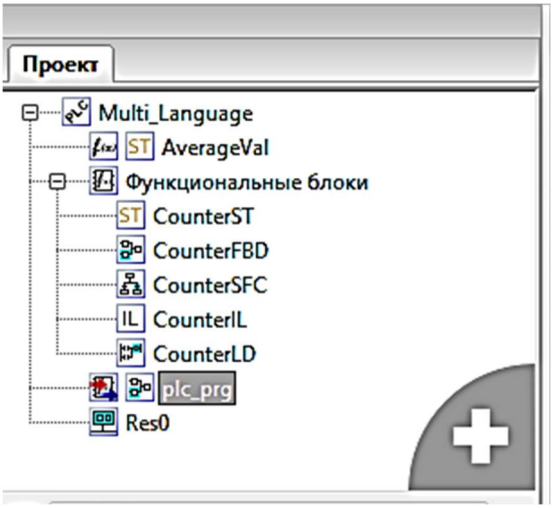


Рисунок 2.1 – Структура проекту Multi Language

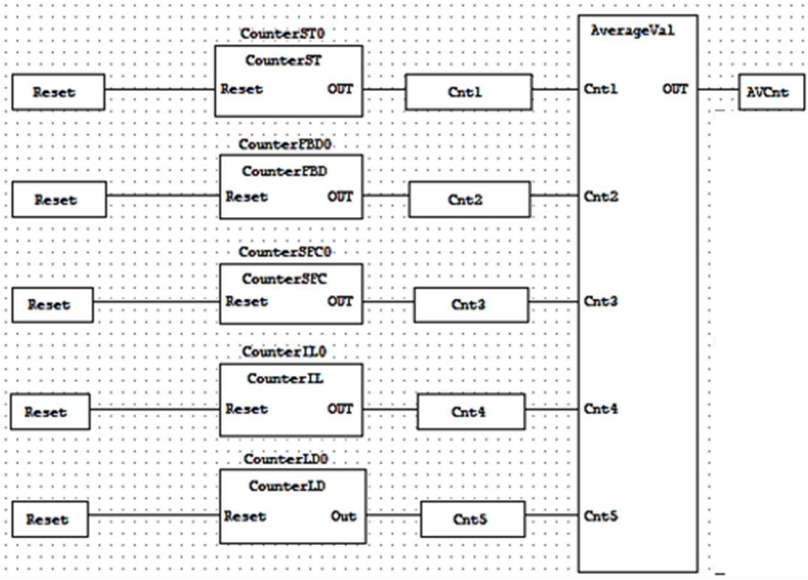


Рисунок 2.2 – Структура програми *plc_prog*

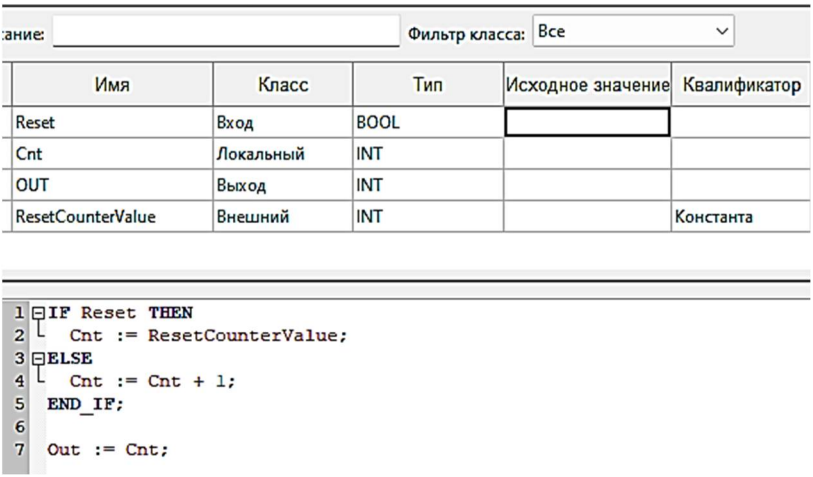
На рис. 2.3 наведено структуру функції *AverageVal* яка безпосередньо обчислює середньоарифметичне значення лічильників. На рис. 2.4 – 2.8 наведено структури функціональних блоків *CounterST, CounterFBD, CounterSFC, CounterIL CounterLD,* відповідно.

Возвращаемый тип: REAL Описание: Фильтр класса: Все

#	Имя	Класс	Тип	Исходное значение	Квалификатор	Описание
1	Cnt1	Вход	INT			
2	Cnt2	Вход	INT			
3	Cnt3	Вход	INT			
4	Cnt4	Вход	INT			
5	Cnt5	Вход	INT			
6	InputsNumber	Локальный	REAL	5.0		Количество входных значений

```
1  AverageVal := INT_TO_REAL(Cnt1+Cnt2+Cnt3+Cnt4+Cnt5)/InputsNumber;
```

Рисунок 2.3 – Структура функції *AverageVal*

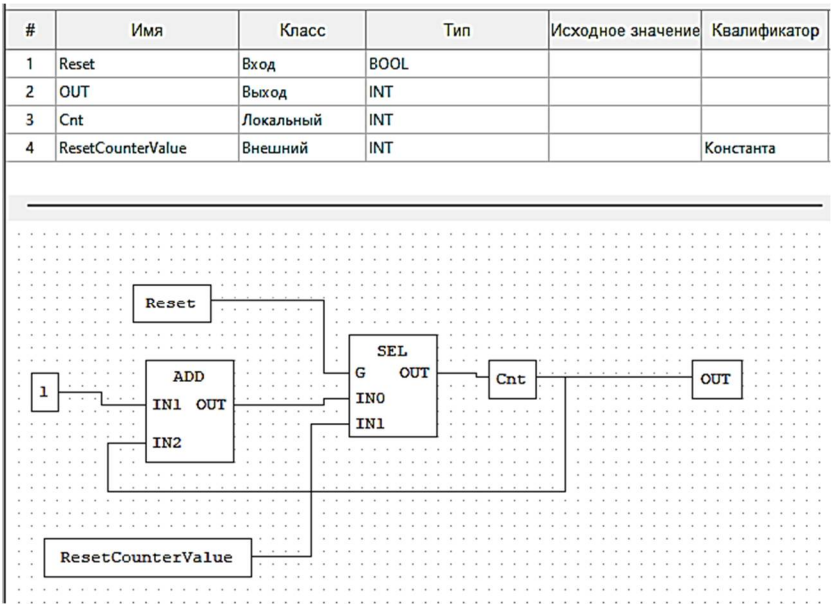


ание: Фильтр класса: Все

	Имя	Класс	Тип	Исходное значение	Квалификатор
	Reset	Вход	BOOL		
	Cnt	Локальный	INT		
	OUT	Выход	INT		
	ResetCounterValue	Внешний	INT		Константа

```
1 IF Reset THEN
2     Cnt := ResetCounterValue;
3 ELSE
4     Cnt := Cnt + 1;
5 END_IF;
6
7 Out := Cnt;
```

Рисунок 2.4 – Структура функціонального блока *CounterST*



#	Имя	Класс	Тип	Исходное значение	Квалификатор
1	Reset	Вход	BOOL		
2	OUT	Выход	INT		
3	Cnt	Локальный	INT		
4	ResetCounterValue	Внешний	INT		Константа

Рисунок 2.5 – Структура функціонального блока *CounterFBD*

	Имя	Класс	Тип	Исходное значение	Квалификатор	
	Reset	Вход	BOOL			
	OUT	Выход	INT			
	Cnt	Локальный	INT			
	ResetCounterValue	Внешний	INT		Константа	

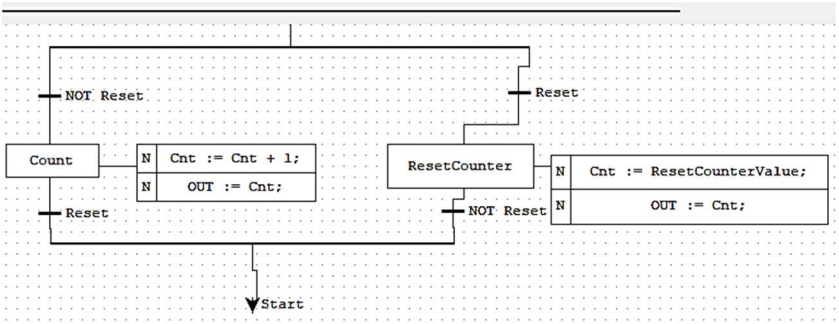


Рисунок 2.6– Структура функціонального блока *CounterSFC*

#	Имя	Класс	Тип	Исходное значение	Квалификатор
1	Cnt	Локальный	INT		
2	Reset	Вход	BOOL		
3	OUT	Выход	INT		
4	ResetCounterValue	Внешний	INT		Константа

```
1  LD Reset
2  JMPC ResetCnt
3
4  (* increment counter *)
5  LD Cnt
6  ADD 1
7  JMP QuitFb
8
9  ResetCnt:
10 (* reset counter *)
11 LD ResetCounterValue
12
13 QuitFb:
14 (* save results *)
15 ST Cnt
16 ST Out
17
```

Рисунок 2.7– Структура функціонального блока *CounterIL*

Имя	Класс	Тип	Исходное значение	Квалификатор
Reset	Вход	BOOL		
Out	Выход	INT		
Cnt	Локальный	INT		
ResetCounterValue	Внешний	INT		Константа

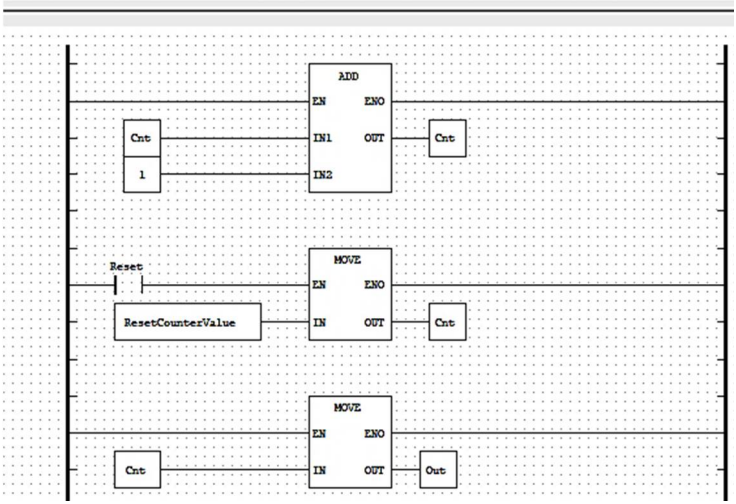


Рисунок 2.8 – Структура функціонального блока *CounterLD*,

Як можна бачити з рис. 2.3 – 2.8, використання функцій і функціональних блоків є ефективним засобом структурування програми.

Хід роботи:

1. Створити проект *plc_prog* мовою FBD у середовищі OpenPLC.
2. У менеджері проекту додати функцію *AverageVal* та описати її структуру як наведено на рис. 2.3.
3. У менеджері проекту додати функціональні блоки *CounterST, CounterFBD, CounterSFC, CounterIL CounterLD* та описати їх структуру.
4. Описати структуру програми plc_prog використовуючи створені функціональні блоки та функцію, які розташовані у

розділі «Пользовательские POU» (Program Organization Unit) у меню блока що додається до програми *plc_prog* (див. рис.2.9).

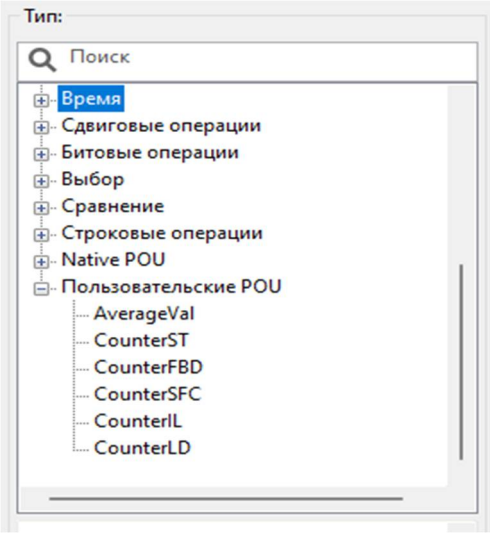


Рисунок 2.9 – Розташування функції та блоків що додаються до програми *plc_prog*

5. Компілювати проект з програмою *plc_prog*, виконати його у налагоджувачі та спостерігати графіки зміни станів лічильників у часі.

Контрольні питання

1. Як додати нові POU до створеного проекту.
2. Чим відрізняється функція від функціонального блоку.
3. Пояснити структуру функції *AverageVal*.
4. Пояснити структуру функціонального блоку. *CounterST*.
5. Пояснити структуру функціонального блоку *CounterFBD*.
6. Пояснити структуру функціонального блоку *CounterSFC*.
7. Пояснити структуру функціонального блоку *CounterIL*.
8. Пояснити структуру функціонального блоку *CounterLD*.

9. Як будувати у середовищі OpenPLC графіки залежності змінних від часу.

Лабораторна робота 3
Програмування плат Ардуіно мовами стандарту МЕК 61131-3

Мета. Вивчити методи завантаження програм створених у середовищі OpenPLC у плати Ардуіно

Короткі теоретичні дані. Традиційно програми управління написані мовою С завантажуються в реальні плати Ардуїно у середовищі пакету Arduino IDE. Для завантаження програм керування, що розроблені у середовищі OpenPLC Editor використовують таблиці відповідності параметрів програми пінам мікроконтролерної плати.

OpenPLC Runtime використовує номенклатуру МЕК 61131-3 для адресації місць введення, виведення та пам'яті. Адресація місць вводу/виводу виконується за допомогою спеціальних символьних послідовностей. Ці послідовності є конкатенацією знаку відсотка «%», префіксу розташування, префіксу розміру та одного або кількох натуральних чисел, розділених пробілами.

Підтримуються такі префікси розташування: • I для введення • O для виведення • M для пам'яті Підтримуються такі префікси розміру: • X для біта (1 біт) • B для байту (8 біт) • W для слова (16 біт) • D для подвійного слова (32 біти) • L для довгого слова (64 біти).

Наприклад, якщо ви хочете прочитати стан першого цифрового введення в змінну BOOL, ви повинні оголосити свою змінну, розташовану за адресою %IX0.0. Якщо ви хочете записати вміст змінної UINT у другий аналоговий вихід, вам слід оголосити свою змінну UINT, розташовану в %QW2.

Відзначимо, що зіставлення ПЛК із фізичним введенням/виведенням залежить від платформи. Бітові (X) адреси ПЛК мають ієрархічну адресу з двох частин. Найменша значуща частина (крайня права) може інтерпретуватися як позиція в байті та має бути в діапазоні від 0 до 7. Найбільша частина (крайня ліва) не повинна перевищувати 1023. Частини

розділяються одним знаком «крапка». Розміри даних, відмінні від X, мають ієрархічну адресу з однієї частини. Вони не повинні містити крапки (.) і не повинні перевищувати максимальну адресу пам'яті для вашої платформи. Відповідність пінів плати Arduino Uno адресам змінних наведено в табл. 3.1

Таблиця 3.1 – Відповідність пінів плати Arduino Uno адресам змінних

Digital In	2, 3, 4, 5, 6	%IX0.0 – %IX0.4
Digital Out	7, 8, 12, 13	%QX0.0 – %QX0.3
Analog In	A0, A1, A2, A3, A4, A5	%IW0 – %IW5
Analog Out	9, 10, 11	%QW0 – %QW2

Обладнання для виконання роботи: плата Arduino Uno R3; USB-кабель; макетна плата; світлодіод, резистор 220 Ом.

Хід роботи:

1. Скласти електричну схему: пін GND Arduino – резистор – катод світлодіоду – світлодіод – пін 7 Arduino.
2. Створити проект мовою LD. Ввести програму наведену на рис. 2.10. Під'єднати Arduino до комп'ютеру кабелем USB.
1. Перевірити працездатність програми за допомогою симулятора
2. Генерувати програму для додатку OpenPLC Runtime для чого натиснути на піктограму з червоною стрілкою.
3. Виконати трансфер програми до PLC (плати Ардуіно) для чого натиснути на піктограму червоне «коло з квадратом» та у конфігураційному вікні (див. рис. 2.11) обрати тип плати та COM порт. Після завантаження спостерігати миготіння світлодіоду, що під'єднано до піну 7 Ардуіно.

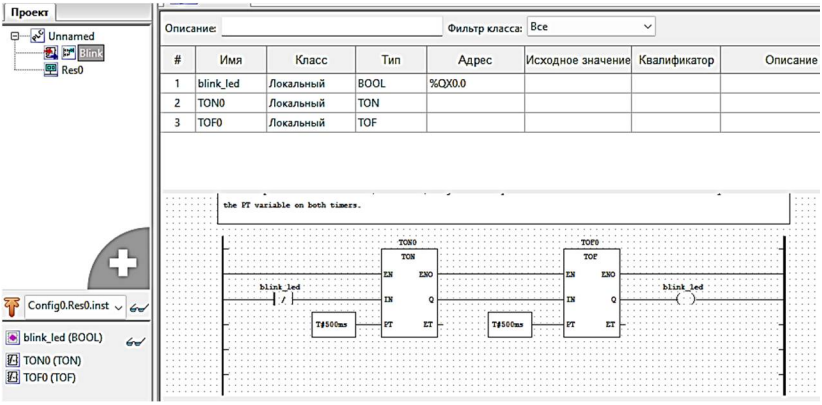


Рисунок 2.10 – Програма миготіння світлодіодом

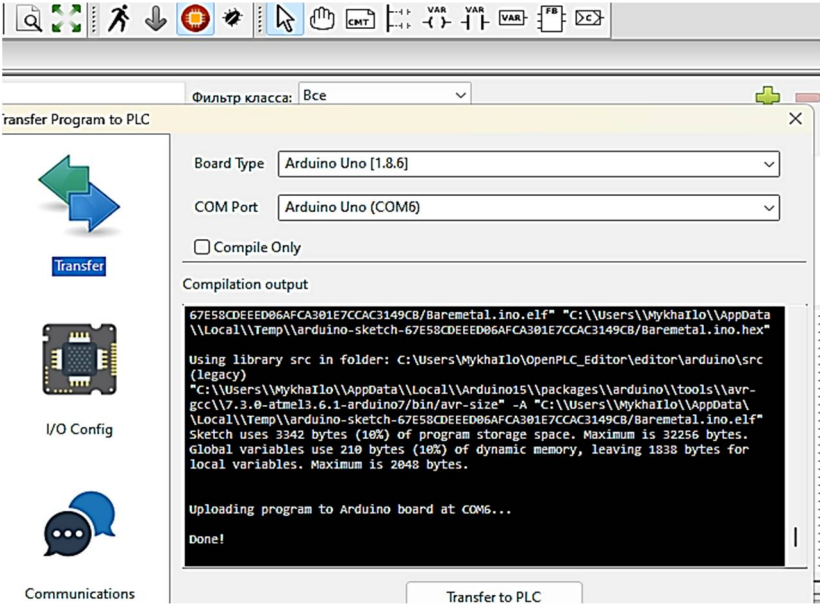


Рисунок 2.11 – Вікно для конфігурування трансферу програми миготіння до PLC

4. Внести зміни у програму (наприклад збільшити частоту миготіння) та повторно виконати пункти 3-5.

5. Розробити програму введення напруги з потенціометра та завантажити її до PLC (плати Ардуіно)

Контрольні запитання

1. Що потрібно для завантаження програми керування у реальний PLC
2. Як розрахувати опір резистора у схемі миготіння.
3. Як прив'язати піни плати Ардуіно до змінних програми.

Лабораторна робота 4
Програмування скінчених автоматів мовами стандарту МЕК 61131-3

Мета. Вивчити методи програмування поведінки керування об'єктом мовами стандарту МЕК 61131-3 використовуючи програми створені у середовищі OpenPLC Editor.

Короткі теоретичні дані. Поведінка додатків керування формалізується засобами теорії скінчених автоматів. Такі автомати мають певну множину входів, виходів, станів, серед яких виділяють початковий стан. Входам та виходам автомату відповідають певні канали контролера (мікроконтролерної плати). На входи контролера надходять сигнали з давачів Виходи під'єднані до виконавчих механізмів. Зміна станів автомату описується функцією переходів, а значення виходів - функцією виходів. Існує два виду автоматів Мілі (англ. Mealy machine) та Мура (англ. Moore machine), які мають різні види цих функцій. Автомату задають за допомогою таблиць або графів. Вершина у графі автомату Мура відповідає певному стану автомату та визначає значення виходів у цьому стані, а дуга описує умови переходу в інший стан. Програма, що реалізує автомат містить ранги(ланцюги) встановлення початкового стану автомату, опису виходів (дій) автомату у кожному стані та опису переходів.

Розглянемо програмування автомата Мура на прикладі. Нехай S_0, S_1, S_2, S_3 – стани автомату; y_0, y_1, y_2, y_3 – дії автомата у відповідних станах; x_0, x_1, x_2, x_3, x_4 – входи автомата;

початковим станом S_f автомата є стан S_2. Функція переходів автомата задана таблицею 4.1.

Таблиця 4.1 – Функція переходів автомата

Звідки	Куди			
	S_0	S_1	S_2	S_3
S_0			x_0	x_3
S_1	x_2			
S_2		x_1		
S_3			x_4	

Граф цього автомата наведено на рис. 2.12, а програму мовою LD – на рис. 2.13.

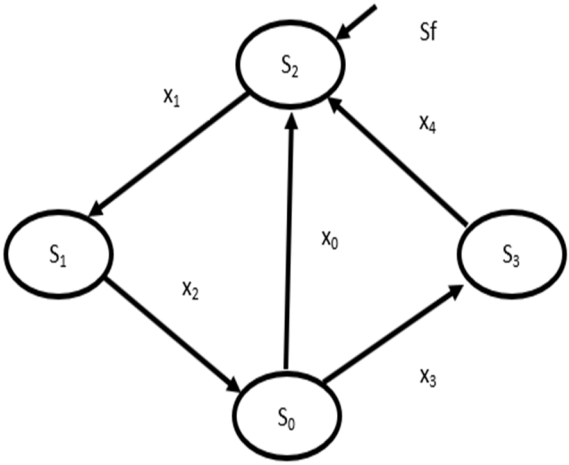


Рисунок 2.12 – Приклад графу скінченого автомату

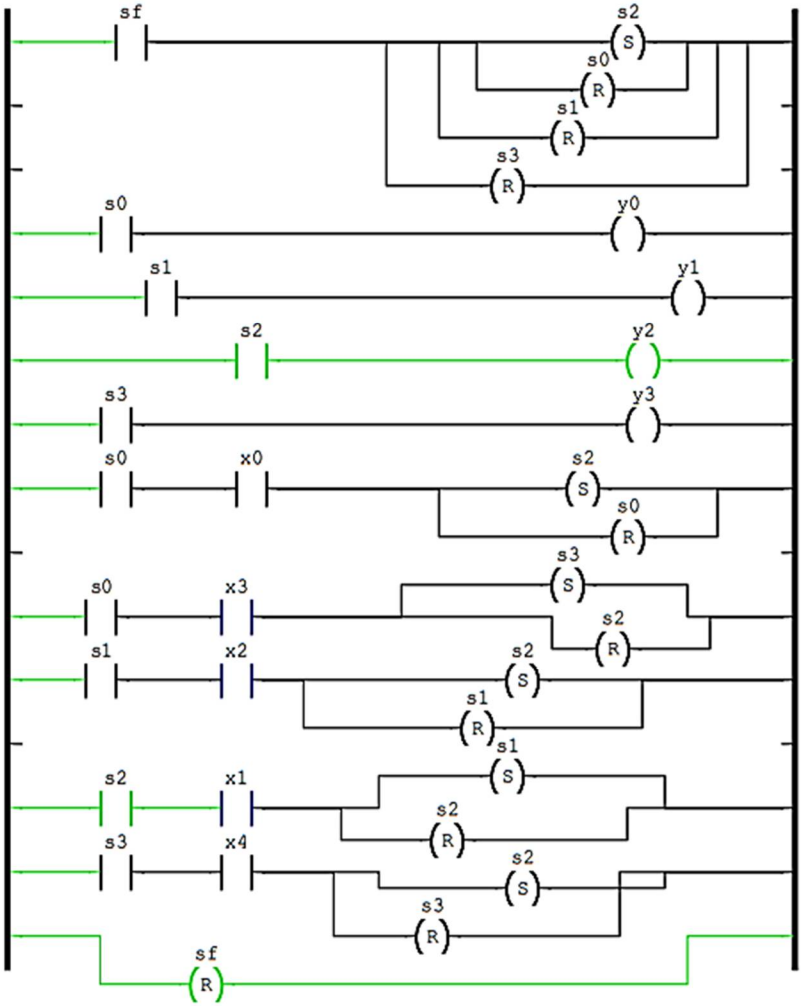


Рисунок 2.13 – Програма що реалізує автомат рис. 2.12.

Хід роботи

1. Набрати програму рис. 2.13 у середовищі OpenPLC Editor та дослідити відповідність її переходів із одного стану в інший графу автомата рис. 2.13.

2. Розробити граф автомату Мура, що реалізує алгоритм керування світлофором на пішохідному переході.

3. Визначити склад контролеру що реалізує алгоритм керування світлофором. Визначити у контролері зміні входів, виходів та станів автомату.

4. Створити у середовищі OpenPLC Editor проект керування мовою FBD або LD або SFC. Конфігурувати змінні контролеру відповідно до вимог попереднього пункту, ввести у програму ранги автомату, запустити створену програму на виконання у середовищі симулятора програми OpenPLC Editor.

5. Протестувати поведінку створеного автомату. Для цього ініціювати зміну станів, шляхом завдання вхідної події «початок переходу вулиці пішоходом». Якщо фактична послідовність зміни станів, або дії у певних станах не відповідають графу автомату, то внести зміни у програму автомату та виконати нове тестування поведінки.

6. Розробити граф автомату Мура, що реалізує алгоритм керування залізничною станцією, яка розташована між пунктами **А** та **В**, має дві колії, приймає потяги з пункту **А** на вільну колію, відправляє потяги за принципом «перший прийшов – перший відправлено». Відправлення потягів відбувається при умові, що колія до пункту **В** вільна. Створити та протестувати додаток керування залізничною станцією.

Контрольні запитання:

1. Визначення, різновиди, множини та функції скінченого автомату.
2. Способи завдання скінченого автомату.
3. Приклади використання скінченого автомату для формування поведінки пристроїв.
4. Принципи побудови програми що реалізує поведінку керуючого автомата

Рекомендовані джерела

1. Віддалений та віртуальний інструментарій в інжинірингу [Текст] : монографія / А. В. Пархоменко, Г. В. Табунщик, М. О. Поляков [та ін.] ; за заг. ред. др. К. Хенке. – Електронні дані. – Запоріжжя : Дике Поле, 2015. – 250 с. – Режим доступу: http://eir.zp.edu.ua/handle/123456789/1970.

2. Remote and virtual tools in engineering [Текст] : textbook / A. V. Parkhomenko, G. V. Tabunshyk, M. O. Poliakov [et al.] ; general editorship Dr.Ing.Karsten Henke. – Zaporizhzhya : Dike Pole, 2016. – 250 p. – Mode of access: http://eir.zp.edu.ua/handle/123456789/1825.