© Kovalenko Yu., Mishyna I., Vyshnevska N., Dzubanenko A. Structure analysis of secured applications programming language for programmable logic controllers // Ukrainian Scientific Journal of Information Security, 2016, vol. 22, issue 1, p. 38-43.

# СТРУКТУРНИЙ АНАЛІЗ МОВ ПРОГРАМУВАННЯ ЗАХИЩЕНИХ ЗАСТОСУНКІВ ДЛЯ ПРОГРАМОВАНИХ ЛОГІЧНИХ КОНТРОЛЕРІВ

## Юлія Коваленко, Ізабелла Мішина, Наталія Вишневська, Андрій Дзюбаненко

*Національний авіаційний університет, Україна*

**КОВАЛЕНКО Юлія Борисівна**, к.пед.н.

*Рік та місце народження:* 1981 рік, м. Баку, Азербайджан.
*Освіта:* Фізико-математичний факультет, Полтавського державного педагогічного університету імені В. Г. Короленка., 2003 рік.
*Посада:* доцент кафедри безпеки інформаційних технологій з 2013 року.
*Наукові інтереси:* об'єктно-орієнтоване програмування, автоматизовані системи управління, захищеність програмного забезпечення, фізика високих енергій (фізика елементарних частинок).
*Публікації:* більше 50 наукових публікацій, серед яких наукові статті, матеріали та тези доповідей на конференціях, навчальні посібники.
*E-mail:* yleejulee22@gmail.com

**МІШИНА Ізабелла Юріївна**

*Рік та місце народження:* 1995 рік, м. Красноармійськ, Україна.
*Освіта:* Національний авіаційний університет, з 2012 року.
*Посада:* студентка.
*Наукові інтереси:* криптографічний захист інформації, розробка захищеного програмного забезпечення.
*Публікації:* декілька доповідей на конференціях.
*E-mail:* bella.mishyna@yandex.ru

**ВИШНЕВСЬКА Наталія Сергіївна**

*Рік та місце народження:* 1977 рік, м. Київ, Україна.
*Освіта:* Київський міжнародний університет цивільної авіації (з 2001 року – Національний авіаційний університет), 2001 рік.
*Посада:* старший кафедри безпеки інформаційних технологій з 2012 року.
*Наукові інтереси:* інформаційна та авіаційна безпека.
*Публікації:* 10 наукових статей.
*E-mail:* viserj@ukr.net

**ДЗЮБАНЕНКО Андрій Васильович**

*Рік та місце народження:* 1983 рік, м. Лубни, Полтавська обл., Україна.
*Освіта:* Національний авіаційний університет, 2007 рік.
*Посада:* здобувач кафедри комп'ютеризованих електротехнічних систем та технологій.
*Наукові інтереси:* алгоритми розпізнавання образів, системи відеоспостереження, нейронні мережі.
*Публікації:* більше 20 наукових публікацій.
*E-mail:* dzubanenko_av@ukr.net

***Анотація.*** *З приходом у сферу промислової автоматизації такого пристрою як програмовані логічні контролери (ПЛК), управління технологічними процесами істотно спростилося. Ведуться розробки нових систем обміну інформаційними даними і нових алгоритмів. Це призводить до величезного різноманіття контролерів. Кожен з них відрізняється специфічними набором функцій, унікальною конструкцією і*

*конкретною мовою управління. У даній статті ми розглядаємо структуру ПЛК, що може допомогти у виборі, а також представляємо структурний аналіз мов програмування ПЛК. Розглянуто стандарт МЕК 61131-3 є забезпечення розробників програм для ПЛК потужними інструментами для підвищення якості застосунків, що включає в себе також надійність та захищеність програм. Проте, у даному стандарті бракує практичних методів перевірки, чи дійсно розроблений застосунок відповідає обраним вимогам з приводу захищеності.*

***Ключові слова:*** *програмований логічний контролер (ПЛК), МЕК 61131-3, мови програмування, система реального часу, мова діаграмного типу SFC.*

**Вступ**

Програмовані логічні контролери (ПЛК) використовуються, здебільшого, у сфері промислової автоматизації для розробки складних управляючих систем. Конструкція цих застосунків має великий вплив з точки зору продуктивності і виробничих витрат. Через складність систем управління і багаторазове використання комбінацій програмно-апаратного забезпечення, розробник має звернути особливу увагу на безпеку цих систем. Одним із шляхів забезпечення безпеки застосунків для ПЛК та підвищення їх якості є використання мов програмування стандарту МЕК 61131-3 [1] [2], який дає можливість розробки незалежних від виробника ПЛК модульних застосунків.

Стандарт МЕК 61131-3 визначає синтаксис і семантику чотирьох мов програмування для ПЛК, а також допоміжний засіб структурування програм (мова діаграмного типу SFC).

До перших чотирьох мов належать: мова релейно-контактних схем LD, мова функціональних діаграм FBD, текстова високорівнева мова ST і текстова мова низького рівня IL. Мова LD, заснована на релейно-контактних схемах, дозволяє описувати логічні функції. Мовою FBD функціональні можливості представляються у вигляді графічних блоків. Текстова мова ST близька до мови Pascal і оперує процедурними, умовними і операторами циклу. IL - апаратно-незалежна ассемблероподобна мова. У той час, як ST і IL є текстовими мовами, а LD і FBD – графічними, SFC, крім власного синтаксису, може бути використана в поєднанні з будь-якою текстовою або графічною мовою.

**Аналіз існуючих досліджень**

У роботі [1] проведено аналіз методів написання захищених програм мовами стандарту МЕК 61131-3, праця [2] містить огляд сукупності методів програмування та деяких алгоритмів, реалізованих та оптимізованих мовами стандарту, що розглядається. Робота [3] містить базові концепти програмування ПЛК, структуру та складові елементи програм ПЛК, опис базового синтаксису мов програмування стандарту МЕК 61131-3. У праці [4] представлений детальний опис мов стандарту, розглянуті ідеї для оптимізації та полегшення процесу розробки програм для ПЛК. У роботі [5] розроблені основні правила створення захищених застосунків для ПЛК, виділено і проаналізовано основні аспекти розробки програм мовами представленого стандарту та мультимовного програмування. [6] є посиланням на 3 частину стандарту МЕК 61131-3:2013, що визначає синтаксис і семантику об'єднаного набору мов програмування ПЛК. У роботі [7] розглядаються основні недоліки та переваги мов стандарту МЕК 61131-3, обговорюються проблеми даного стандарту та можливі альтернативи. У праці [8] викладені базові знання про психологію програмування і властивості аналізованих мов, необхідні для вибору мови реалізації для будь-якого проекту.

**Основна частина дослідження**

Ми розглядаємо програмований логічний контролер (ПЛК) як пристрій, що виконує управління фізичними процесами по записаному в ньому алгоритму, орієнтований на роботу з пристроями через розвинене введення сигналів датчиків і виведення сигналів на виконуючі механізми. ПЛК призначені для роботи в системах реального часу. Відзначимо, що однією з переваг систем ПЛК є модульність, тобто, можливість комбінування і поєднання видів пристроїв введення-виведення таким чином, який найбільш підходить даному застосунку.

Структура ПЛК може бути розділена на 4 частини: модулі введення-виведення, центральний процесор, пам'ять і термінал (рис. 1).

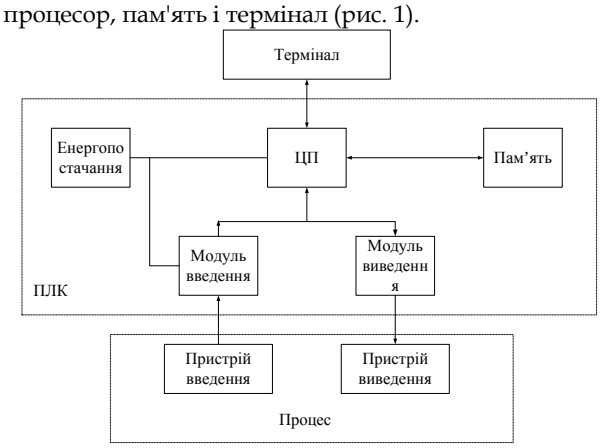


Рис. 1. Структура ПЛК

Виділимо основні принципи мов МЕК 61131-3:

1) Вся програма розбивається на безліч функціональних елементів - Program Organization Units (POU). Ці елементи можуть бути реалізовані за допомогою мов стандарту і складаються з трьох категорій [3]:

- функції POU під час виконання повертають тільки одне значення, що визначається одним з типів даного стандарту, результат функції і довільну множину додаткових вихідних даних. Ці функціональні елементи не мають певного стану (не містять ніякої інформації про стан), тобто виклик функції з одними і тими ж аргументами завжди дає один і той же результат;

- функціональні блоки POU повертають одне або більше значень в якості результату. Стан функціонального блоку зберігається від одного виконання до наступного - володіє імовірнісною післядією - тому виклик з одними і тими ж аргументами може повернути різні результати;

- програми POU можна визначити як «логічне з'єднання всіх елементів і конструкцій мов програмування, необхідних для обробки вибраного тону для управління механізмом або процесом за допомогою програмованого контролера» (IEC, 2003). Їх визначення і використання еквівалентно функціональним блокам. Вони також можуть використовувати дві попередні категорії POU в якості додаткових елементів.

2) стандарт вимагає суворої типізації даних. Вказівка типів даних дозволяє виявити більшість помилок в програмі до її виконання;

3) є засоби для виконання різних фрагментів програми в різний час, з різною швидкістю, а також паралельно. Наприклад, один фрагмент програми може сканувати кінцевий датчик з частотою 100 разів на секунду, в той час як другий фрагмент буде сканувати датчик температури з частотою один раз в 10 секунд;

4) для виконання операцій в певній послідовності, яка задається моментами часу або подіями, використовується спеціальна мова послідовних функціональних схем (SFC);

5) стандарт підтримує структури для опису різнорідних даних. Наприклад, температуру підшипників насоса, тиск і стан «включено-виключено» можна описати за допомогою єдиної структури «Pomp» і передавати її всередині програми як єдиний елемент даних;

6) стандарт забезпечує спільне використання всіх п'яти мов, тому для кожного фрагмента завдання може бути обрана будь-яка, найбільш зручна, мова;

7) програма, написана для одного контролера, може бути перенесена на будь-який контролер, сумісний зі стандартом МЕК 61131-3 [4].

Отже, проведемо аналіз п'яти мов стандарту МЕК 61131-3 для ПЛК. Для використання в програмі ПЛК кожної з мов необхідно зробити два кроки [5]:

1) визначити семантику, засновану на МЕК 61131-3;

2) визначити системи переходів, що представляють елементи мови, а для ST і LD - визначити загальні правила компонування елементів в програму. Це приводить до визначення операційної семантики.

Операційна семантика складається з однієї системи переходів, що моделює цикл виконання ПЛК, плюс одна система переходів для кожної мови, що представляє поведінкові правила вибраної мови і декількох систем переходів, що моделюють елементи програми. Моделювання поведінкових правил залежить від обраної мови.

Структура мови SFC перетворюється в системі переходів, яка розраховує алгоритм зміни програми. Підхід LD і ST мов вдаліший при компіляції програми в системі переходів.

IL (Instruction List) - текстова мова низького рівня. Вона універсальна і часто використовується як загальна проміжна мову, на яку переводяться інші мови. IL - лінійно-орієнтована мова, її основною перевагою є простота вивчення, IL можна програмувати за допомогою будь-якого текстового редактора. Найчастіше використовується для вирішення невеликих завдань з малою кількістю розгалужень і написання найбільш критичних місць в програмі, тому що вона дозволяє створювати високоефективні і оптимізовані функції.

На момент третьої редакції стандарту МЕК 61131-3 IL оголошена застарілою і небажаною для використання [6].

У мові ST використаний інший підхід. ST (Structured Text) - мова високого рівня, що містить безліч конструкцій для присвоєння значень змінним, виконання функцій і функціональних блоків, написаних виразів, умовних переходів, вибору операторів, побудови ітераційних процесів. Ця мова призначена в основному для виконання складних математичних обчислень, опису складних функцій, функціональних блоків і програм. Її переваги над IL - це дуже стисле формулювання завдань програмування, зрозуміла конструкція програми в блоках операторів і основних конструкціях для управління потоком виконання. Алгоритм ST ділиться на кілька кроків (операторів), які використовуються для обчислення та присвоєння значень з метою управління потоком виконання та виклику або виходу з POU. На відміну від IL, оператор в ST може визначатися кількома рядками або більшою кількістю операторів, які можуть бути записані в одному рядку.

Мова релейно-контактних схем LD (Ladder Diagram) заснована на релейно-контактних логічних блок-схемах з горизонтальними ланками між вертикальними шинами електроживлення, які виконуються послідовно. Ця мова програмування розроблена, в головній мірі, для обробки булевих сигналів (істина / неправда). Шини пов'язують LD-мережу зліва і справа. Від лівої шини, під керуванням стану сигналу «1» струм досягає всі підключені елементи. Залежно від їх стану, елементи або дозволяють току пройти до наступних елементів, або переривають потік.

LD-мережа описується вертикальними і горизонтальними лініями, а також точками перетину. Контакт виконує логічну операцію, грунтуючись на значенні з вхідної лінії і значенні необхідної змінної. Тип логічної операції залежить від типу контакту. Значення, що отримується з правої підключеної лінії, є шуканим результатом.

Однак мову LD проблематично використовувати для реалізації складних алгоритмів, тому що вона не підтримує підпрограми, функції, інкапсуляцію і інші засоби структурування програм з метою підвищення якості програмування. Ці недоліки ускладнюють багаторазове використання програмних компонентів, що робить програму довгою і складною для обслуговування. Недоліком є також те, що лише невелика частина програми вміщається на моніторі комп'ютера або панелі оператора при програмуванні.

Незважаючи на зазначені недоліки, мова LD відноситься до найбільш поширених у світі [3], хоча

використовується для програмування тільки простих завдань.

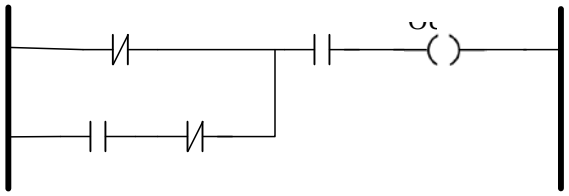

Рис. 2. Приклад програми на мові LD

FBD (Functional Block Diagram) - графічна мова програмування, яка використовує функціональні структурні схеми для представлення прикладної програми для ПЛК. Вона заснована на принципових схемах, обумовленому графічному поданні електричних ланцюгів і базових елементів мови - блоків. Блок - це підпрограма, функція або функціональний блок. Функціональні блоки інкапсулюють дані і методи, чим нагадують об'єктно-орієнтовані мови програмування, але не підтримують успадкування і поліморфізм. Ще однією особливістю блоків є параметризація числа входів і виходів, які можуть бути представлені різними типами. Наприклад, блок функції ADD може мати від двох до довільно допустимої кількості входів, а на вихід буде поданий результат підсумовування всіх входів.

FBD запозичує символіку булевої алгебри і, так як булеві символи мають входи і виходи, які можуть бути з'єднані між собою, FBD більш ефективна для подання структурної інформації, ніж мова релейно-контактних схем. Також крім основних для графічних мов елементів, FBD має в наявності деякі власні елементи.

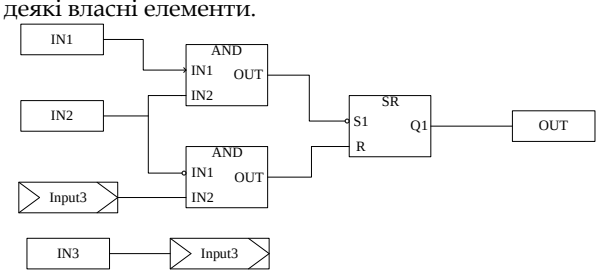


Рис. 3. Приклад програми на мові FBD

Мова діаграмного типу SFC (Sequential Function Chart) дозволяє представити програму POU або функціональний блок ПЛК за допомогою графічної і текстової системи позначень. SFC по суті є допоміжним засобом для структурування програм за допомогою розбиття основної керуючої гілки застосунку на менші компоненти і контролю їх виконання. Основою цієї мови є математичний апарат мереж Петрі (СП), що дозволяє описати процеси в формі дводольних орієнтованих графів [7]. Графічне представлення дозволяє чітко позначити потік виконання програми, а також робить можливим проектування послідовних і паралельних процесів застосунку.

Спрацювання будь-якого переходу Tj в розміченій мережі веде до зміни розмітки.

Виконання менших програмних компонент (наприклад, процесів або гілок) залежить як від умов, визначених програмою, так і від поведінки вхідних-вихідних даних. Компоненти безпосередньо програмуються на одній з інших мов стандарту МЕК 61131-3. Процеси з покроковою поведінкою особливо підходять для програмування на SFC.

Перший рівень структурування на SFC - мережа, яка складається з елементів під назвою кроки і стани. Крок може бути активним або неактивним. Коли він активний, необхідні команди виконуються до тих пір, поки крок не стане неактивним. Заміну статусу кроку визначає стан переходу, який є логічним виразом. Якщо умова переходу стає «Істина», то наступний крок стає активним, а попередній крок дезактивується.

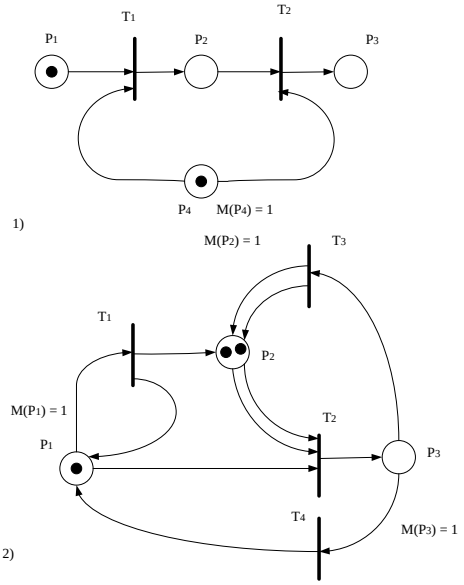


Рис. 4. Подання процесів у вигляді дводольних графів. Тут: Pi - позиції, Tj - переходи, M (Pi) - розмітка

Зі зміною переходу, властивість «активний» переходить від активного кроку до його наступника або наступників; таке пересування утворює мережу. Ця властивість може бути розділена в разі виконання паралельних гілок, а потім відновлена при закінченні виконання таких гілок.

Дії в кроках мають спеціальні класифікатори, що визначають спосіб їх виконання всередині кроку: циклічне виконання (N), одноразове виконання (P) і т.д. Всього таких класифікаторів дев'ять, причому серед них є класифікатори зі збереженими (S), відкладеними (D) і обмеженими за часом (L) діями.

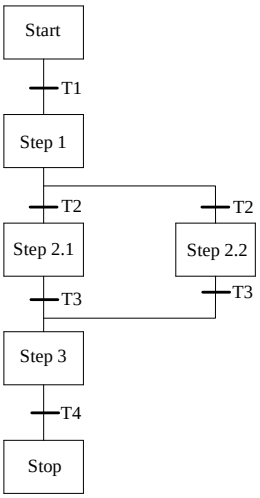


Рис. 5. Набір кроків мережі SFC, об'єднаних переходами

У праці [8] представлені базові знання з психології програмування та властивостях мов, які мають бути виділені при виборі мови програмування для конкретної задачі. Керуючись даною роботою, для наведених мов програмування можемо виділити загальні критерії, які є критичними при програмуванні ПЛК. До таких віднесемо: легкість у вивченні, універсальність, циклічність, логічний паралелізм, наявність операцій з аналоговими сигналами, наявність обмежень на складність алгоритму, наявність механізмів синхронізації, структуризації, абстрагування та подієвість. Сутність деяких з наведених критеріїв відображена у праці [9].

Циклічність управляючого алгоритму може бути обумовлена за схемою: зчитування стану вихідних сигналів через датчики – їх обробка і формування вихідних сигналів – видача вихідних сигналів на виконуючі органи. Логічний паралелізм відображає існування множини процесів, що паралельно протікають в об'єкті управління, і передбачає наявність в алгоритмі управління незалежних або слабо залежних частин – логічно відокремлених потоків управління. Наявність механізмів синхронізації передбачає відповідність виконання алгоритму управління фізичним процесам у зовнішньому середовищі, що обумовлює необхідність розвиненої служби часу і активні роботу з часовими об'єктами: затримками, паузами, таймаутами. Механізми структуризації у даному контексті означають мовні засоби організації спільного функціонування логічно паралельних частин, а механізми абстрагування – понятійний перехід від датчиків і виконуючих органів до цільового технологічного процесу. Подієвість передбачає алгоритмічні зміни програми і набору оброблюваних нею вхідних/вихідних сигналів в залежності від подій, що відбуваються всередині об'єкта.

Однією із основних цілей стандарту МЕК 61131-3 є забезпечення розробників програм для ПЛК потужними інструментами для підвищення якості застосунків, що включає в себе також надійність та захищеність програм. Проте, у даному стандарті бракує практичних методів перевірки, чи дійсно розроблений застосунок відповідає обраним вимогам з приводу захищеності. Тому, окрім наявних у стандарті методів підвищення надійності та захищеності програм, більша частина реалізації відповідності застосунків цим критеріям лягає на плечі розробників.

У таблиці 1 представлена відповідність мов стандарту МЕК 61131-3 визначеним критеріям. Позначка «+/-» означає, що відповідність даному критерію може бути забезпечена для програм, написаних даними мовами, але є дуже складною у реалізації. У таблиці розглядаються мови LD, FBD та поєднання SFC+ST. SFC+IL у роботі [9] не рекомендується для використання, а в третій частині стандарту МЕК 61131 мова IL оголошена застарілою [6], тому у даному дослідженні вона не береться до уваги.

Таблиця 1

Критеріальний аналіз методів програмування за вибраним стандартом

| **Критерій** | **LD** | **FBD** | **SFC+ST** |
|---|---|---|---|
| Легкість у вивченні | + | + | + |
| Універсальність | - | - | + |
| Циклічність | + | +/- | +/- |
| Логічний паралелізм | + | + | + |
| Операції з аналоговими сигналами | +/- | + | + |
| Обмеження на складність алгоритму | + | + | + |
| Синхронізація | +/- | + | + |
| Структуризація | +/- | + | +/- |
| Абстрагування | - | - | +/- |
| Подієвість | - | - | + |
| Надійність | + | +/- | +/- |
| Захищеність | + | +/- | + |

**Висновки**

Провівши аналіз мов програмування, приходимо до висновку: керуючі системи, які працюють в логічних процесах реального часу, можуть бути запрограмовані на SFC+ ST. Мова ST головним чином може бути використана для створення модулів програмного забезпечення з математичним контекстом, наприклад, для опису алгоритмів управління. LD, FBD і IL - мови програмування, які підходять для формулювання основних дій і підходять для керуючих систем, які можуть описані простими логічними діями або логічними сигналами.

Поєднання SFC + ST є найбільш, мабуть, потужним і універсальним засобом зі складу мов МЕК 61131-3. Графіка мови SFC полегшує вивчення мови, наявність спільних коренів з мережами Петрі частково знімає проблеми синхронізації і паралелізму. Програмування операцій з аналоговими і логічними змінними є досить комфортним за рахунок використання текстової Паскаль-подібної мови ST. Управління потоком команд не викликає проблем. Суттєва перевага підходу – подієвість, природним чином підтримувана через механізм «крок-перехід». Відладка програм може бути полегшена візуальним трасуванням потоку управління. Слабкі місця мови SFC (як і мереж Петрі) - це абстрагування і структурування [7], що, як і в попередніх випадках, негативно позначається на складності програмованих алгоритмів і їх якості (супроводжуваності і надійності). У порівнянні з LD і FBD підхід SFC+ST програє в зручності програмування паралелізму алгоритму і, відповідно, більш підходить для програмування лінійних алгоритмічних послідовностей.

Мова чи сукупність мов стандарту МЕК 61131-3 мають підбиратися відповідно до цільової задачі розроблюваної системи, критерії реалізації якої можуть бути виділені і співставлені із критеріями, визначеними у даній статті.

**Література**

[1] Bonfatti F., Monari P.D. and Sampieri U. IEC 1131-3 programming methodology. Software engineering methods for industrial automated systems, CJ International Editions, ISBN 2-9511585-0-5.

[2] Ohman M., Johansson S. and Arzиn K.E., Implementation aspects of the PLC standard IEC 1131-3, IFAC Control Engineering Practice 123. – Vol. 6. – №°4. – P. 547-555

[3] Barbosa H., Dйharbe D. Formal Verification of PLC Programs Using the B Method. - Proceedings of the Third international conference on Abstract State Machines, Alloy, B, VDM, and Z, 2012. – P. 353-356

[4] Lewis R.W. Programming industrial control systems using IEC 113-3 Revised edition. – The Institution of Electrical Engineers, London, UK, 1998. – 329 p.

[5] De Smet O., Couffin S., Rossi O., Canet G., Lesage J.-J., Schnoebelen Ph., Papini H. Safe programming of PLC using formal verification methods, Ecole Normale Suprieure, Chaire De Fabrications, France, 2000.

[6] IEC 61131-3:2013 Programmable controllers - Part 3: Programming languages.

[7] Анисимов Н.А., Голенков Е.А., Харитонов Д.И. Композиционный подход к разработке параллельных и распределенных систем на основе сетей Петри // Программирование. – 2001. –№6.

[8] Зюбин В.Е. Графика или текст: какой язык нужен программисту? // Открытые системы. – 2004. – №1.

[9] Зюбин В.Е. Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. – 2005. – №11. – С. 31-35.

**УДК 004.056.5 (045)**

***Коваленко Ю.Б., Мишина И.Ю., Вишневская Н.С., Дзюбаненко А.В. Структурный анализ языков программирования защищенных приложений для программируемых логических контроллеров***

***Аннотация.*** *С приходом в сферу промышленной автоматизации такого устройства как программируемые логические контроллеры (ПЛК), управление технологическими процессами существенно упростилось. Ведутся разработки новых систем обмена информационными данными и новых алгоритмов. Это приводит к огромному многообразию контроллеров. Каждый из них отличается специфическим набором функций, уникальной конструкцией и конкретным языком управления. В данной статье мы рассматриваем структуру ПЛК, которая может помочь в выборе, а также представляем структурный анализ языков программирования ПЛК. Рассмотрены стандарт МЭК 61131-3 является обеспечение разработчиков программ для ПЛК мощными инструментами для повышения качества приложений, включает в себя также надежность и защищенность программ. Однако, в данном стандарте не хватает практических методов проверки, действительно разработан приложение отвечает выбранным требованиям по поводу защищенности.*

***Ключевые слова:*** *программируемый логический контроллер (ПЛК), МЭК 61131-3, языки программирования, система реального времени, язык диаграммного типа SFC.*

***Kovalenko Yu., Mishyna I., Vyshnevska N., Dzubanenko A. Structure analysis of secured applications programming language for programmable logic controllers***

***Abstract.*** *Application of programmable logic controller in the sphere of industrial automation substantially simplified technological processes control. New data exchange systems and new algorithms are being developed. It leads to enormous variety of comptrollers. Each of them differs by specific set of functions, unique construction and certain control language. In this article we describe structure of PLC, that can help to choose one, and also present the structural analysis of PLC programming languages. IEC 61131-3 standard considered is to provide software developers PLC powerful tools to improve the quality of applications that includes the safety and security applications. However, this lack of standard practices check whether the designed application correspond to the requirements regarding protection.*

***Key words:*** *programmable logic controller (PLC), IEC 61131-3, programming languages, real-time system, diagram type language SFC.*

Отримано 2 *лютого 2016 року*, затверджено редколегією 2 *березня 2016 року*