УДК 681.5

СУЧАСНІ ЗАСОБИ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОМИСЛОВИХ КОНТРОЛЕРІВ В СЕРЕДОВИЩІ CODESYS

Ківа І. Л., Поночевний Є. О., Лісовець С. М.

Київський національний університет технологій та дизайну

Розглянуто сучасні засоби програмування промислових контролерів в середовищі CoDeSys на текстових (IL і ST) та графічних (LD, FBD і SFC) мовах програмування. Виконано аналіз існуючих засобів програмування промислових контролерів, а також експериментальну перевірку різних мов програмування промислових контролерів.

Ключові слова: *дія, інженерний інтерфейс, засіб візуалізації, засіб програмування, мова програмування, програма, промисловий контролер, процесор, середовище виконання програмного коду, функція, функціональний блок*

Покращення якості продукції, яка виробляється різними галузями промисловості України, зазвичай передбачає або застосування більш складних технологічних процесів, або підвищення точності контролю параметрів для вже існуючих технологічних процесів. І в першому, і в другому випадках алгоритми обробки отриманої інформації і формування керуючих впливів ускладняються, що вимагає застосування засобів обчислювальної техніки.

Типовим засобом обчислювальної техніки, який застосовується для автоматизації технологічних процесів, є промисловий контролер. Окрім безпосередньо цифрових компонентів, таких як процесор, Flash-пам'ять, EEPROM-пам'ять тощо, він утримує також спеціалізовані компоненти, характерні лише для засобів автоматизації: аналого-цифрові та цифро-аналогові перетворювачі, аналогові компаратори, послідовні інтерфейси RS-232 і RS-485 тощо. Природно, що для програмування промислових контролерів необхідно застосовувати відповідні засоби програмування – одним з таких засобів є середовище програмування CoDeSys (CoDeSys утворюється від слів Controllers Development System) фірми 3S-Smart Software Solutions.

CoDeSys надає програмісту зручне середовище для програмування контролерів на мовах програмування, які описані в стандарті IEC 61131-3:2013. Воно підтримує п'ять основних мов програмування, до яких відносяться текстові мови програмування (Instruction List (IL) і Structured Text (ST)) та графічні мови програмування (Ladder Diagram (LD), Function Block Diagram (FBD) і Sequential Function Chart (SFC)).

Необхідно зауважити, що середовище CoDeSys утримує не тільки безпосередньо мови програмування, а також ряд додаткових засобів підтримки промислових контролерів, таких як інженерний інтерфейс CoDeSys ENI Server, засоби візуалізації CoDeSys HMI, середовище виконання програмного коду CoDeSys SP RTE, засоби керування рухом CoDeSys SoftMotion і багато чого «корисного».

Задачі керування потребують неперервного циклічного контролю. Під час одного робочого циклу промисловий контролер здійснює як основні операції (читання стану входів, виконання програмного коду, запис стану виходів), так і додаткові операції (обслуговування апаратних ресурсів, моніторинг системи виконання, контроль часу робочого циклу, перехід на початок робочого циклу). Відповідно, розв'язуючи задачу створення програмного забезпечення для певного технологічного процесу, принципово можна користуватися будь-якою з мов програмування. Але, в залежності від технологічного процесу, в одному випадку може підходити одна мова програмування, в іншому випадку – інша мова програмування. CoDeSys допускає одночасне використання в одному проекті кількох мов програмування, але зазвичай в якості основної застосовується тільки якась одна з них, а інші застосовуються як допоміжні для написання функцій, функціональних блоків, програм та дій. Так як вибір мови програмування, яка найбільш підходить для певного технологічного процесу, зменшує загальну кількість помилок і недоробок в програмному коді, то таким чином підвищується надійність програмного коду. А надійність програмного коду, в свою чергу, дозволяє зменшити аварійність певного технологічного процесу.

Постановка завдання

Завдання полягало в підвищенні надійності генеруємого програмного коду, який застосовується для програмування промислового контролера. Для цього було визначено відповідність між типом технологічного процесу і різновидом мови програмування промислових контролерів, при якій забезпечується найбільша надійність генеруємого програмного коду, і при цьому зберігаються такі характеристики програмного коду, як швидкодія, компактність і читабельність. Також необхідно було розробити критерії оцінки мов програмування промислових контролерів, за якими можна в залежності від того або іншого технологічного процесу рекомендувати ту або іншу мову програмування. А також провести експериментальну перевірку різних мов програмування промислових контролерів для одного і того ж

технологічного процесу і виконати їх порівняння для оцінки надійності, швидкодії, компактності і читабельності програмного коду.

Результати досліджень

Як відомо, мова програмування Instruction List (IL) дослівно – список інструкцій. Це типовий асемблер з акумулятором і переходами по міткам. Набір інструкцій стандартизований і не залежить від конкретної цільової платформи. IL – сама проста в реалізації мова програмування. В складі МЕК-мов програмування мова IL застосовується при створенні компактних компонентів, які потребують старанного пророоблення, на яке не жалко часу.

При роботі з IL набагато адекватніше, ніж при роботі з іншими мовами, можна представити, як буде виглядати відтрансльований код. Завдяки цьому мова програмування IL виграє там, де необхідно досягнути найбільшої ефективності. До компіляторів цей відноситься в повній мірі. В системах виконання з інтерпретатором проміжного коду виграш не настільки помітний. В режимі виконання середовище програмування CoDeSys автоматично перетворює вікно редактора у вікно моніторингу. Значення всіх змінних відображаються справа від команд IL і доступні для зміни. Можна встановити або скинути точку зупинки і прокрокувати програму по одній команді.

Цікаві можливості для IL надає режим Flow Control середовища програмування CoDeSys. В цьому режимі у вікні моніторингу підсвічуються номера рядків, які виконувалися в попередньому робочому циклі, і відображаються відповідні значення акумулятора.

Основою ST-програм є вирази. Результат обчислення виразів присвоюється змінній за допомогою оператора «:=», як в мові програмування Паскаль. Кожний вираз обов'язково завершується символом «;». Вираз будується із змінних, констант і функцій, розділених операторами. Стандартні оператори в виразах ST мають символьне представлення, наприклад, математичні дії: «+», «–», «*», «/», операції порівняння тощо. Окрім операторів, елементи виразу можна відокремлювати пробілами і табуляціями для кращого сприйняття. В текст можуть бути введені коментарі. Усюди, де допустимі пасивні роздільники, можна вставляти і коментарі. Декілька виразів можна записати підряд в один рядок.

Але гарним стилем вважається запис одного виразу в рядку. Довгі вирази можна перенести на наступний рядок. Перенесення рядка рівноцінно пасивному роздільнику.

Вираз може утримувати інший вираз, який повинен знаходитися в скобках. Вираз, що знаходиться в скобках, обчислюється в першу чергу. Тип виразу визначається типом результату обчислень.

LD-схема представляє собою дві вертикальні шини живлення, між якими розташовані горизонтальні ланцюги, які утворені контактами і обмотками реле. Кількість контактів в ланцюгу може бути довільною, але реле завжди одне. Якщо послідовно з'єднані контакти замкнуті, струм іде по ланцюгу і реле вмикається. За необхідності можна підключати паралельно декілька реле, але послідовне підключення реле не допускається.

В LD-схемі кожному контакту ставиться у відповідність логічна змінна, яка визначає його стан. Якщо контакт замкнутий, то змінна має значення TRUE. Якщо ж контакт розімкнутий, то змінна має значення FALSE. Ідентифікатор змінної пишеться над контактом і фактично є його найменуванням. Послідовне з'єднання контактів або ланцюгів рівноцінно логічній операції І. Паралельне з'єднання утворює логічну операцію АБО.

Ланцюг може бути або замкнутий (ON), або розімкнутий (OFF). Стан ланцюга як раз і відображається на обмотці реле і відповідно на значенні логічної змінної цієї обмотки (TRUE або FALSE).

Схема FBD будується з компонентів, які на ній відображаються в вигляді прямокутників. Входи POU відображаються зліва від прямокутника, виходи POU – справа. Всередині прямокутника вказується тип POU і найменування входів і виходів. Для екземпляра функціонального блока його найменування вказується зверху, над прямокутником. Прямокутник компонента може утримувати рисунок, який відображає тип цього компонента. Розмір прямокутника залежить від кількості входів і виходів та встановлюється графічним редактором автоматично.

Програма в FBD не обов'язково повинна представляти єдину велику схему. FBD-схема може утворюватися з багатьох ланцюгів, які виконуються один за одним. В CoDeSys всі ланцюги одного POU, пронумеровані і розділені горизонтальними лініями, відображаються в єдиному графічному вікні. Значення змінних, які обчислені в одному ланцюгу, доступні в наступних ланцюгах зразу ж в тому ж самому робочому циклі.

Виконання FBD-ланцюгів здійснюється зліва направо, зверху вниз. FBD-блоки, які розташовані лівіше, виконуються раніше. FBD-блок починає обчислюватися тільки після обчислення значень всіх його входів. Подальші обчислення не будуть продовжені до

завершення обчислень на всіх виходах цього FBD-блоку. Іншими словами, значення на всіх виходах FBD-блоку з'являються одночасно. Обчислення ланцюга вважається закінченим тільки після обчислення значень на виходах всіх елементів, які входять до його складу. В CoDeSys редактор FBD автоматично розставляє блоки в порядку їх виконання.

Будь-яка SFC-схема складається з елементів, які представляють собою кроки і умови переходів. Кроки показані на SFC-схемі в вигляді прямокутників. Реальна робота кроку (дії) описується в окремому вікні середовища програмування CoDeSys і не відображається на SFC-схемі. Про призначення SFC-кроку повідомляє тільки його назва або, якщо назви недостатньо, короткий текстовий опис (коментар). Кроки на SFC-схемі можуть бути пустими, що не викликає похибки при компіляції проекту. Пусті кроки є нормою при застосуванні програмування зверху донизу, яке характерне для мови програмування SFC. Визначити дії, які відповідають кроку, можна в будь-який час.

Нижче кроку на з'єднувальній лінії присутня горизонтальна риска, яка позначає перехід. Умовою переходу можуть бути: логічна змінна, логічний вираз, константа або пряма адреса. Перехід здійснюється при виконанні двох умов: перехід дозволений (його відповідний крок активний) або умова переходу має значення TRUE. Прості умови відображаються безпосередньо на SFC-схемі справа від риски, яка позначає перехід. Для громіздких умов застосовується інший підхід.

Кожна SFC-схема починається з кроку, який виділений графічно подвійними вертикальними лініями або по всьому периметру. Це так знаний початковий крок. Найменування початкового кроку може бути довільним (за замовчуванням найменування початкового кроку Init). Початковий крок присутній обов'язково, хоча і може бути пустим. Декілька гілок SFC-схеми можуть бути паралельними. Ознакою паралельних гілок на SFC-схемі є подвійна горизонтальна лінія. Кожна паралельна гілка починається і закінчується кроком. Тобто умова входу в паралельність завжди одна на всіх, а умова виходу з паралельності теж одна на всіх. Декілька гілок SFC-схеми можуть бути альтернативними. Ознакою альтернативності гілок на SFC-схемі є одинарна горизонтальна лінія. Кожна альтернативна гілка починається і закінчується власною умовою переходу. Перевірка альтернативних умов починається зліва направо. Якщо правдива умова знайдена, то інші альтернативи не розглядаються. В альтернативних гілках завжди виконується тільки одна з гілок, тому її завершення і буде означати перехід до наступного за альтернативною групою кроку.

В загальному випадку SFC-схема виконується зверху донизу. Стандартом допускається створення переходів на довільний крок. Для цього застосовуються з'єднувальні лінії з проміжними стрілками або поіменовані переходи. Тобто перехід здійснюється на крок, ім'я якого вказано під стрілкою. В англомовній літературі перехід на довільний крок має назву «стрибок» (jump). В різних системах програмування для налагодження SFC-схем застосовуються наступні прийоми: анімація активних кроків і дій, примусове встановлення активних кроків, блокування перевірки або фіксація умов переходів, пропуск виконання заданих кроків, блокування виконання дій, моніторинг часу активності кроків. Загальним методом відслідковування роботи SFC-схеми в реальному масштабі часу є моніторинг і трасування змінних, кроків і дій, спеціально створених допоміжних логічних прапорів і лічильників активності. Розширені налагоджувальні функції реалізуються в системах програмування різними способами і залежать від системи виконання.

Яка з мов програмування МЕК дозволяє створювати найбільш компактний код? Відповідь на це питання досить проста: розмір коду практично не залежить від мови реалізації. Переведення програми з однієї мови на іншу не має ніякого смислу з точки зору оптимізації.

Застосування міток і переходів в графічних мовах програмування дає можливість формального переведення розгалужених ST-програм і IL-програм на LD-схеми, FBD-схеми і SFC-схеми. Програмний код, який генерується в цьому випадку, практично співпадає незалежно від мови програмування (за виключенням мови програмування SFC). Це пов'язано з тим, що мова програмування SFC найбільш наближена (у порівнянні з іншими мовами програмування) до мов програмування високого рівня. Відповідно, для перевірки переходів і керування активністю кроків є потреба в ефективному механізмі керування, який створює додатковий код. Але при цьому мова програмування SFC є універсальною та наглядною, і дозволяє швидко реалізовувати складні проекти. Отримати більш компактний машинний код можна тільки при ручній переробці програмних алгоритмів. При цьому використання різних мов програмування МЕК потребує різного підходу і різного мислення. Тому ефективність програмного коду, в кінцевому випадку, залежить виключно від якості пророблення цього програмного коду і від ступеня володіння тією або іншою мовою програмування.

Висновки

Встановлено, що для певного технологічного процесу все ж таки можна рекомендувати застосування певної мови програмування промислових контролерів, зменшуючи таким чином аварійність цього технологічного процесу і трохи підвищуючи надійність, швидкодію, компактність і читабельність програмного коду.

ЛІТЕРАТУРА

1. Петров И. В. Программируемые контроллеры. Практическое применение языков стандарта МЭК 61131-3 [Текст] / И. В. Петров / Под ред. проф. В. П. Дьяконова. – М. : Солон-Пресс, 2004. – 254 с.
2. Карпов Ю. Г. Теория автоматов [Текст] / Ю. Г. Карпов. – СПб. : Питер, 2002. – 224 с.
3. Шалыто А. А. Логическое управление. Методы аппаратной и программной реализации алгоритмов [Текст] / А. А. Шалыто. – СПб. : Наука, 2000. – 780 с.
4. Дьяконов В. П. Компьютерная математика. Теория и практика [Текст] / В. П. Дьяконов. – М. : Нолидж, 2001. – 1296 с.
5. Мозговой М. В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход [Текст] / М. В. Мозговой. – СПб. : Наука и Техника, 2006. – 320 с.
6. Филлипс Ч., Харбор Р. Системы управления с обратной связью [Текст] / Ч. Филлипс, Р. Харбор. – М. : Лаборатория Базовых Знаний, 2001. – 616 с.

Современные средства создания программного обеспечения для промышленных контроллеров в среде CoDeSys

Кива И. Л., Поночевный Е. А., Лисовец С. Н.
Киевский национальный университет технологий и дизайна

Рассмотрено современные средства программирования промышленных контроллеров в среде CoDeSys на текстовых (IL и ST) и графических (LD, FBD и SFC) языках программирования. Выполнен анализ существующих средств программирования промышленных контроллеров, а также экспериментальную проверку разных языков программирования промышленных контроллеров.

Ключевые слова: *действие, инженерный интерфейс, средство визуализации, средство программирования, язык программирования, программа, промышленный контроллер, процессор, среда выполнения программного кода, функция, функциональный блок*

Modern tools for creating software for industrial controllers in the environment CoDeSys

Kiva I. L., Ponochevniy E. A., Lisovets S. N.
Kyiv national university of technologies and design

The modern tools of programming of industrial controllers with CoDeSys environment in the text (IL and ST) and graphical (LD, FBD and SFC) programming languages. It performs analysis of existing means of programming industrial controllers, as well as experimental verification of different programming languages for industrial controllers.

Keywords: *act, engineering interface, renderer, software, programming language, program, industrial controller, processor, runtime code, function, functional block*