Міністерство освіти і науки України
Тернопільський національний технічний університет
ім. Івана Пулюя

Факультет прикладних інформаційних технологій
та електроінженерії
Кафедра автоматизації технологічних
процесів і виробництв

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи №22
«Модифікація та відлагодження проекту у середовищі програмного забезпечення «Factory I/O»-CODESYS»

з курсу «Проектування систем автоматизації »

для студентів спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Тернопіль, 2018

Методичні вказівки до лабораторної роботи № 22 на тему «Модифікація та відлагодження проекту у середовищі програмного забезпечення «Factory I/O»-COSESYS» з курсу «Проектування систем автоматизації» / Шкодзінський О.К., Пісьціо В.П., Сікора Д.А., Герасимів Ю.О. - Тернопіль: ТНТУ, 2018 - 16 с.
Для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології.

Укладачі: Шкодзінський О.К., Пісьціо В.П., Сікора Д.А., Герасимів Ю.О.

Методичні вказівки розглянуті, схвалені і затверджені на засіданні кафедри автоматизації технологічних процесів та виробництв Тернопільського національного технічного університету імені Івана Пулюя
(протокол № 11 від 26 лютого 2018 року).

Тема: Модифікація та відлагодження проекту у середовищі програмного забезпечення «Factory I/O»-CODESYS.

Мета: Ознайомитись з принципами програмування мовою **«Structured text»** (**ST**) - «Структурований текст» за стандартом **МЕК 61131-3.** Навчитися вносити зміни у проекти «Factory I/O»-CODESYS та відлагоджувати їх на коректну роботу.

1.ТЕОРЕТИЧНІ ВІДОМОСТІ

У попередній лабораторній роботі було розглянуто готовий проект для програмного середовища «Factory I/O» та підключення його 3D-моделі до віртуального контролера. Керування конвеєром здійснювалось програмно, програма написана мовою **ST** (Structured text) у середовищі CODESYS.

Основою ST-програми слугують вирази. Результат обчислення виразу присвоюють змінній за допомогою оператора «:=». Кожний вираз обов'язково має закінчуватись крапкою з комою «;». Вирази будуються із змінних, констант і функцій, розділених операторами.

Стандартні оператори у виразах мають символьне позначення, наприклад математичні дії: +, -, *, /, порівняння тощо. Крім операторів у виразі можуть використовуватись пробіли і табуляція для кращого сприйняття, а також, коментарі.

Вираз може містити інший вираз у дужках, що обчислюється у першу чергу. Вираз обчислюється відповідно до правил пріоритету операцій. У наступній таблиці наданий список ST-операторів, розташованих у порядку пріоритету.

Таблиця 1 Пріоритет операторів

№	Операція	Позначення	Пріоритет
1	Вираження в дужках	(Вираження)	Найвищий
2	Виклик функції	**Ім'я функції (список параметрів)**	1
3	Зведення в ступінь	**EXPT**	2
4	Заміна знаку	–	3
5	Числове доповнення	**NOT**	4
6	Розподіл	/	5
7	Залишок від розподілу	**MOD**	6
8	Додавання	+	7
9	Віднімання	–	8
10	Порівняння	<, >, <=, >=	9
11	Нерівність	<>	10
12	Рівність	=	11
13	Логічне **І**	**AND**	12
14	Логічне, що виключає **АБО**	**XOR**	13
15	Логічне **АБО**	**OR**	Найнижчий

1.1. Основні оператори мови ST

Оператор вибору (IF)

Оператор вибору дозволяє виконувати різні групи виразів залежно від умов, записаних логічними виразами. Повний синтаксис оператора **IF** має вигляд:

```
IF <логічний вираз IF>
THEN
<вирази IF>;
[
ELSEIF <логічний вираз ELSEIF 1>
THEN
<вирази ELSEIF 1>;
…
ELSEIF <логічний вираз ELSEIF n>
THEN
<вирази ELSEIF n>;
ELSE
<вирази ELSE>;
]
END_IF
```

Якщо **<**логічний вираз **IF> TRUE (ІСТИНА)**, то виконуються вирази першої групи – <вирази **IF**>. Інші вирази пропускаються, альтернативні умови не перевіряються. Застосовувати оператор присвоювання **«:=»** в <логічному виразі **IF**> **не можна**.

Частина конструкції у квадратних дужках є необов'язковою й може бути відсутня.

Якщо **<**логічний вираз **IF> FALSE (НЕПРАВДА)**, то одна за одною перевіряються умови **ELSIF**. Перша істинна умова приведе до виконання відповідної групи виразів. Інші умови **ELSIF** аналізуватися не будуть. Груп **ELSIF** може бути декілька або не бути зовсім.

Якщо всі логічні вирази дали помилковий результат, то виконуються вирази групи **ELSE**, якщо вона є. Якщо групи **ELSE** нема, то не виконується нічого.

Оператор множинного вибору (CASE)

Оператор множинного вибору **CASE** дозволяє виконати різні групи виразів залежно від значення однієї цілочисельної змінної або виразу.

Синтаксис:

```
CASE <цілочисельний вираз> OF
<значення 1>:
<вирази 1>;
<значення 2>:
<вирази 2>;
…
<значення n>:
<вирази n>
[
ELSE
<вирази ELSE>;
]
END_CASE
```

Якщо значення виразу збігається із заданою константою, то виконується відповідна група виразів. Інші умови не аналізуються **(**<значення **1**>**:** <вирази **1**> **;)**.

Якщо декілька значень констант повинні відповідати одній групі виразів, їх можна перелічити через кому **(**<значення **2**> **,** <значення **3**> **:** <вирази **3**> **;)**. Діапазон значень можна визначити через двокрапку **(**<значення **4**>**:** <значення **5**> **:** <вирази **4**> **;)**.

Група виразів **ELSE** є необов'язковою. Вона виконується за незбігу жодної з умов **(**<вирази **ELSE**> **;)**.

Цикли WHILE та REPEAT

Цикли **WHILE** та **REPEAT** забезпечують повторення групи виразів, поки вірним є логічний вираз. Якщо умовний вираз завжди істина, то цикл стає нескінченним.

Умова в циклі **WHILE** перевіряється до початку циклу. Якщо логічний вираз первісно має значення **FALSE (НЕПРАВДА)**, тіло циклу не буде виконане жодного разу.

Синтаксис:

```
WHILE <умовний логічний вираз> DO
<вирази (тіло циклу)>
END_WHILE
```

Умова в циклі **REPEAT** перевіряється після виконання тіла циклу. Якщо логічний вираз первісно має значення **FALSE (НЕПРАВДА)**, тіло циклу буде виконано один раз.

Синтаксис:

```
REPEAT
<вирази (тіло циклу)>
UNTIL <умовний логічний вираз>
END_REPEAT
```

Правильно побудований цикл **WHILE** або **REPEAT** обов'язково повинен виконувати зміну змінних, які складають умову закінчення в тілі циклу, поступово наближаючись до умови завершення. Якщо цього не зробити, цикл не закінчиться ніколи.

Слід намагатися не використовувати точну рівність і нерівність для припинення циклу. Інакше існує ймовірність помилково проскочити граничні умови. Краще використовувати умови більше й менше.

Для реалізації мінімального часу виконання циклу необхідно уникати в тілі циклу й в умовному виразі обчислень, які можна було зробити заздалегідь. Такі обчислення повторюються в циклі, щоразу забираючи час.

Цикл **REPEAT** відрізняється від циклу **WHILE** тим, що перша перевірка умови виходу із циклу здійснюється, коли цикл уже виконався **1** раз. Це означає, що незалежно від умови виходу цикл виконується хоча б один раз.

Зауваження. Програміст повинен бути впевнений, що цикл не стане нескінченним. Для цього в тілі циклу значення змінної, яка визначає вхідну умову, обов'язково повинна змінюватися. Наприклад, шляхом інкремента або декремента лічильника.

Цикл FOR

Цикл **FOR** забезпечує задану кількість повторень групи виразів.

Синтаксис:

```
FOR <цілочисельний лічильник> := <початкове значення>
TO <кінцеве значення>
[BY <крок>] DO
<вирази (тіло циклу)>
END_FOR
```

Перед виконанням циклу лічильник одержує початкове значення. Далі тіло циклу повторюється, поки значення лічильника не перевищить кінцевого значення. Лічильник збільшується в кожному циклі. Початкове й кінцеве значення й крок можуть бути як константами, так і виразами.

Лічильник змінюється після виконання тіла циклу. Тому якщо задати кінцеве значення менше від початкового, то за позитивного збільшення цикл не буде виконаний жодного разу. За однакових початкового й кінцевого значень тіло циклу буде виконано один раз.

Частина конструкції **BY** у дужках необов'язкова, вона визначає крок збільшення лічильника. За замовчуванням лічильник збільшується на одиницю в кожній ітерації. Як лічильник можна використати змінну будь-якого цілого типу.

1.2. Оператори переривання ітерацій EXIT та RETURN

Оператор **EXIT**, який розташований у тілі циклів **WHILE**, **REPEAT** й **FOR**, приводить до негайного закінчення циклу. Оператор **RETURN** здійснює негайний вихід з програми. Розглянемо, наприклад, пошук елемента масиву з певним значенням (**x)**. Найпростіше виконати лінійний перебір за допомогою циклу **FOR**:

Синтаксис:

```
Element:= FALSE;
FOR i := 1 TO Maxlndex DO
IF x - a[i] THEN
Index := i;
Element := TRUE;
EXIT;
END_IF
```

```
END_FOR
IF Element THEN (*елемент знайдено, його індекс – Index*)
```

2. ВІДЛАГОДЖЕННЯ ПРОГРАМИ КЕРУВАННЯ

2.1. Відображення стану датчиків і приводів

Кожен датчик або привід має один або декілька станів. Стани використовуються для зв'язування значень виконавчих механізмів та датчиків з контролером. Крім того, стани також можуть бути використані для управління виконавчими механізмами вручну та аналізу виконання програми.

Стан описується іменем та значенням. При створенні елементу імена автоматично призначаються станам. Зазвичай краще переназвати стани короткими і описовими іменами, тому що вони будуть використовуватися при відображенні виконавчих елементів та датчиків у віртуальному середовищі. Значення можуть бути трьох різних типів даних, залежно від типу та конфігурації датчиків та приводів: **Boolean** (логічне значення) для станів ввімкнення/вимкнення, **Float** для аналогових значень (дійсних чисел) та **Integer** для цілочисельних даних. Кожен тип даних має свій колір. Відповідно можна динамічно відслідковувати яка змінна якого типу даних у вікні поєднання тегів датчиків та виконавчих пристроїв (рис.1).

Рис.1. Типи змінних та їх кольори

Можна показувати або приховувати відображення станів датчиків (Sensors Tags) та виконавчих приводів (Actuators Tags) через натискання відповідних піктограм на панелі інструментів (рис.2).

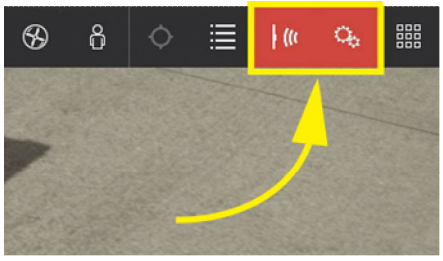

Рис.2. Кнопки включення/виключення відображення станів датчиків і виконавчих механізмів

Відображення станів можна прикріпити у верхньому лівому куті вікна,

натиснувши на відповідних датчиках чи виконавчих механізмах лівою кнопкою миші. Ці стани будуть відображатись постійно, незалежно від положення камери огляду (рис.3). Крім того, ці стани можна перейменовувати та перемикати вручну для виявлення помилок у програмі при відлагодженні. Щоб сховати всі прикріплені відображення станів, слід натиснути **View** > **Clear Docked Tags**.

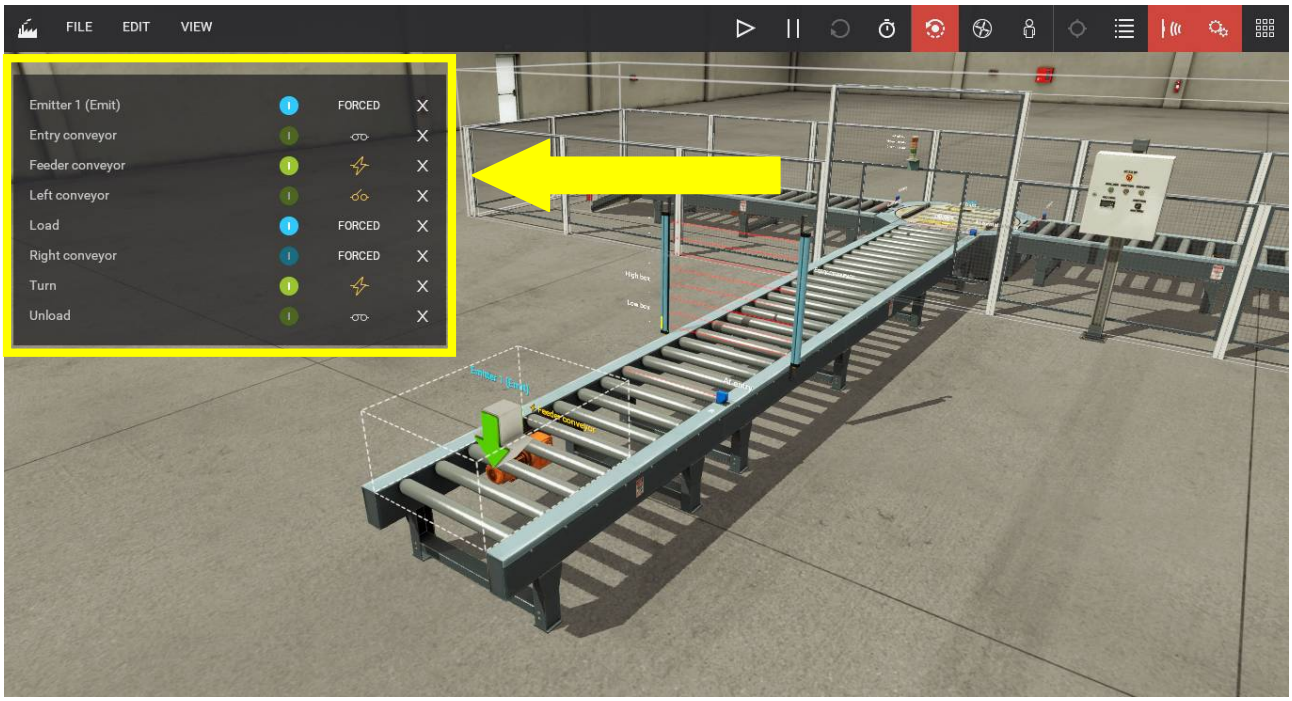

Рис. 3. Відображення станів окремих компонентів сцени

2.2. Примусова зміна стану датчиків і приводів

Стани датчиків і виконавчих механізмів можуть бути примусово змінені натисканням **ЛКМ** (лівої клавіші мишки) на кнопці, слайдері або введенні у полі стану (залежно від типу даних). При натисканні кнопки виконавчого механізму, можна перевизначити значення його стану, отримане з контролера. Примусово змінюючи стани виконавчих елементів, можна вручну виконувати роль контролера та керувати роботою обладнання. Натисканням на кнопки датчиків можна перезадати їх стан і відправити відповідний сигнал на контролер незалежно від стану обладнання.

Таблиця 2. Зміни стану різних типів

Тип	Значення ON	Значення OFF	Зміна величини
Логічний (Bool)			**ЛКМ** для перемикання on/off.
Дійсне число (Float)			**ЛКМ** і пересування для встановлення потрібного значеня.
Цілочисельний (Integer)	4	**0**	**ЛКМ** і ввести нове значення.

Актуалізація примусової зміни стану здійснюється натисненням **ЛКМ** кнопки **RELEASE.** Стани компонентів сцени можуть бути в автоматичному (зелений кружечок) та в ручному режимі (синій кружечок) (рис. 4).

Рис. 4. Стани компонентів сцени

2.3. Моделювання зміни станів

Кожна сцена включає в себе чотири вбудовані стани (рис.5), які контролер може використати для отримання релевантних даних про моделювання (симуляцію).

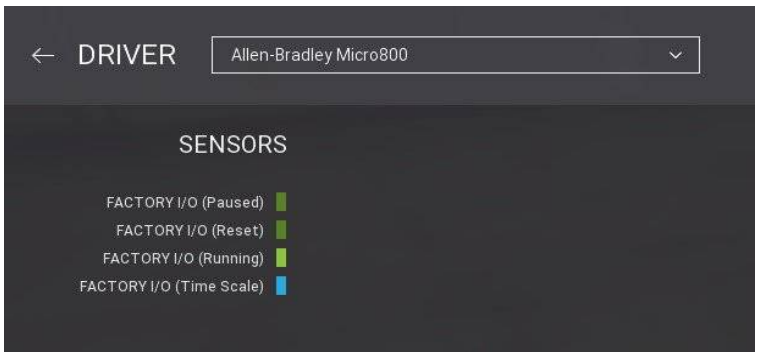

Рис. 5. Вбудовані стани

Таблиця. 3. Стани симуляції Factory I/O

Стан симуляції	Опис
FACTORY I/O (Paused)	**True**, коли симулювання роботи є на паузі
FACTORY I/O (Reset)	**True** (протягом 1с), перезапуск симулювання
FACTORY I/O (Running)	Коли **True,** симулювання відбувається
FACTORY I/O (Time Scale)	Масштабування симуляції у часі: • Режим паузи або редагування = 0 • Сповільнене виконання = 0.1 • Виконання у реальному часі = 1

3. ЗАВДАННЯ

Здійснити модифікацію базового проекту (який розглядався у лабораторній роботі №21), поданого у Додатку А, згідно заданого викладачем варіанту. Варіанти завдань можна взяти з таблиці, наведеній нижче. Слід налагодити сцену на коректну роботу за допомогою внесення змін у програмний код, написаний мовою ST.

Таблиця 2 Варіанти завдань

Варіант	Завдання
1	Сортування коробок на палетах за висотою (високі рухаються прямо, низькі повертають праворуч)
2	Почергове подаванн коробок на палетах по 2 штуки у гілки конвеєра
3	Подавання коробок на палетах почергово: 2 штуки прямо, 1 праворуч
4	Подавання коробок на палетах прямо у послідовності одна висока, одна низька, інші коробки направляються праворуч
5	Подавання коробок на палетах у послідовності 1 штука прямо, 2 штуки праворуч
6	Сортування коробок на палетах за висотою (низькі рухаються прямо, високі повертають праворуч)

4. ЗМІСТ ЗВІТУ

1) На титульній сторінці креслярським шрифтом вказати назву закладу, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище та ініціали викладача, який керував роботою, рік виконання роботи.
2) Вказати тему та мету роботи.
3) Продемонструвати виконану роботу за допомогою скріншотів.
4) Подати текст програми, доповнивши його коментарями.
5) Написати висновок до лабораторної роботи.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1) Поясніть різницю між циклом WHILE і REPEAT.
2) Як працює інструкція IF та яке її призначення?
3) Який формат даних має лічильник циклів?
4) Для чого призначений цикл FOR?
5) У яких режимах масштабу часу може працювати симулятор роботи?

6. ПЕРЕЛІК ПОСИЛАНЬ

1. Методичні вказівки до лабораторної роботи № 21 на тему «Ознайомлення з основами роботи у середовищі програмного забезпечення «Factory I/O» та запуск готового проекту» з курсу «Проектування систем автоматизації» / Шкодзінський О.К., Пісьціо В.П., Сікора Д.А., Герасимів Ю.О. - Тернопіль: ТНТУ, 2018 - 20 с.
2. IEC 61131-3:2013 Programmable controllers — Part 3: Programming languages.
3. Веб-сторінка фірми Smart Software Solutions Gmb виробника середовища CoDeSys. Режим доступу: http://www.3s-software.com/
4. Веб-сторінка ПК "Пролог", підтримка середовища CoDeSys російською мовою. Режим доступу: http://www.codesys.ru/
5. About FACTORY I/O [Електронний ресурс] / NEXT-GEN PLC TRAINING 3D FACTORY SIMULATION. Режим доступу: https://factoryio.com/docs/
6. Навчальні ролики на YouTube. Режим доступу: https://www.youtube.com/watch?v=avemCOBn5lc&t=88s
7. Петров И.В Программируемые контроллеры. Стандартные языки и приемы прикладного программирования / Под ред. проф. В.П. Дьяконова. – М: ООО «СОЛОН-Пресс», 2004. – 256 с.

Зміст

Текст програми

```
PROGRAM PLC_PRG
VAR
	State:BYTE:=0;
	h:BOOL;
	Number:INT:=0;
	T1:TON;
END_VAR

T1(IN := FIO.iAtLoadPos , PT:= T#1S);

IF FIO.kReset THEN state:=0;
END_IF

CASE state OF
	0:
	FIO.oEntryConveyor:=TRUE;
	FIO.oLoad := TRUE;
	FIO.oTurn:=FALSE;
	FIO.oUnload:=FALSE;

	IF FIO.iLowBox THEN state:=1;
	END_IF

	1:
	IF  T1.Q
	THEN state:= 2;
	END_IF

	2:
	FIO.oEntryConveyor:=FALSE;
	IF FIO.iAtUnloadPos THEN state:=3;
	IF  Number=0 THEN Number:=1;
	ELSE Number:=0;
	END_IF
	END_IF

	3:
	FIO.oLoad:=FALSE;
	IF NUMBER=0 THEN State:=4;
	ELSE state :=5;
	END_IF

	4:
	FIO.oLoad:=TRUE;
	IF FIO.iAtUnloadPos=FALSE THEN state:=0;
	END_IF

	5:
	FIO.oTurn:=TRUE;
	IF FIO.iLimit90=TRUE THEN state:=6;
	END_IF
```

```
      6:
      FIO.oUnload:=TRUE;
      IF FIO.iAtLoadPos=TRUE THEN state:=7;
      END_IF

      7:
      IF FIO.iAtLoadPos=FALSE THEN state:=8;
      END_IF

      8:
      FIO.oTurn:=FALSE;
      FIO.oUnload:=FALSE;
      IF Fio.iLimit0 THEN STATE:=0;
      END_IF

END_CASE
```