Міністерство освіти і науки України

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ім. О.С. ПОПОВА

Кафедра комп'ютерно-інтегрованих технологічних процесів і виробництв

Методичні вказівки до виконання лабораторної роботи "Програмування ПЛК. Вивчення мови лінійних інструкцій (IL)"

з дисципліни
"Мікропроцесорні та програмні засоби автоматизації"

Модуль 1 Керування процесом за допомогою цифрових пристроїв у засобах автоматизації.
Побудова систем управління на контролерах та мікроконтролерах різних типів

для бакалаврів галузі знань 15
"Автоматизація та приладобудування"
за спеціальністю 151
"Автоматизація та комп'ютерно-інтегровані технології"

Одеса 2017

УДК 681.5.004.2:004.4 План НМВ 2017 р.

Рецензент – д.т.н., проф. Лисовий І.П.

Тігарєв А.М. Методичні вказівки до виконання лабораторної роботи “Програмування ПЛК. Вивчення мови лінійних інструкцій (IL)” / Тігарєв А.М. – Одеса: ОНАЗ ім. О.С. Попова, 2017. – 16 с.

У методичних вказівках розглянуто реалізацію автоматизації ділянок технологічного процесу за допомогою мови лінійних інструкцій (IL). Указано вимоги до виконання програм мовою програмування IL за стандартом МЕК 61131-3.

Ухвалено
на кафедрі комп’ютерно-інтегрованих
технологічних процесів і виробництв
і рекомендовано до друку
Протокол № 4
від 10.11.2017 р.

Затверджено
методичною радою
академії зв’язку
Протокол № 5
від 28.02.2017 р.

Підписано до друку 26.06.2017.
Формат 60/88/16 Обсяг 0,9 друк. арк.
Тираж 25 прим. Зам. № 08/17.
Віддруковано в редакційно-видавничому центрі ОНАЗ ім. О.С. Попова
м. Одеса, вул. Ковалевського, 5
Тел. 7050 494

ЗМІСТ

Лабораторна робота № 4

ПРОГРАМУВАННЯ ПЛК. ВИВЧЕННЯ МОВИ IL

1 Мета роботи

Метою роботи є ознайомлення із принципами програмування мовою лінійних інструкцій (**IL**) за стандартом МЕК 61131-3 для програмованих логічних контролерів при розробці окремих вузлів систем керування, а також набуття навичок роботи в інтегрованому середовищі розробки програм CoDeSys.

2 Основні положення

Будь-який технологічний процес повинен виконуватися відповідно до певних правил, послідовностей операцій, тобто відповідно до його "АЛГОРИТМУ ФУНКЦІОНУВАННЯ" та "АЛГОРИТМУ КЕРУВАННЯ".

Складні процеси можуть бути розділені на елементарні операції, взаємодія яких здійснюється за елементарною логікою. При автоматизації технологічних процесів виникає необхідність у керуванні процесами без безпосереднього втручання людини, в цьому випадку спеціальні технічні засоби автоматизації повинні забезпечити ведення процесу відповідно до його алгоритму функціонування.

2.1 Мова лінійних інструкцій IL

Мова **IL** (Instruction list) дослівно – список інструкцій. Це типовий асемблер з акумулятором і переходами по мітках. Набір інструкцій стандартизований і не залежить від конкретної цільової платформи. Оскільки IL найпростіша у реалізації мова, вона отримала значне поширення до прийняття стандарту МЕК. Точніше, не сама **IL**, а дуже схожі на неї реалізації. Практично всі виробники ПЛК Європи створювали подібні системи програмування, схожі на сучасну мову **IL**. Найбільший вплив на формування сучасного **IL** зробила мова програмування **STEP** контролерів фірми Siemens. Мова **IL** дозволяє працювати з будь-якими типами даних, викликати функції й функціональні блоки, реалізовані будь-якою мовою. Таким чином, на **IL** можна реалізувати алгоритм будь-якої складності, хоча текст буде досить громіздким.

У складі МЕК-мов **IL** застосовується при створенні компактних компонентів, що вимагають ретельного пророблення. При роботі з **IL** набагато краще, ніж з іншими мовами, можна уявити, як буде виглядати відтрансльований код. Завдяки чому, **IL** виграє там, де потрібно досягти найвищої ефективності. Особливо це відноситься до компіляторів. У системах виконання з інтерпретатором проміжного коду виграш не настільки значний.

2.1.1 Формат інструкції

Текст на **IL** – це текстовий список послідовних інструкцій. Кожна інструкція записується на окремому рядку й містить оператори, в залежності від типу операції, один і більше операндів, розділених комами. Інструкція може включати 4 поля, розділені пробілами або знаками табуляції:

Мітка: Оператор Операнд Коментар

Перед операндом може знаходитися мітка, що закінчується двокрапкою (:). Мітка інструкції не є обов'язковою, вона ставиться тільки там, де потрібно. Оператор присутній обов'язково. Операнд необхідний майже завжди. Коментар – необов'язкове поле, записується наприкінці рядка. Коментар записується між двома круглими дужками та символами (* *). Коментар повинен бути останнім елементом у рядку. Між інструкціями можуть бути порожні рядки.

Давати коментарі між полями інструкції не можна.

Приклад **IL**-програми:

```
МІТКА1:     LD      Sync  (*приклад IL*)
            AND     Start
            SQ
(*мітку можна поставити в окремий рядок*)
МІТКА2:
            LD      2     (*y = 2 + 2*)
            ADD     2
            ST      y
```

Для кращого сприйняття рядка **IL** вирівнюють звичайно у стовпчики відносно поля.

Редактор CoDeSys вирівнює текст автоматично. Крім цього, редактор «нальоту» виконує синтаксичний контроль і виділення кольором. Так, коректно введені оператори виділяються блакитним кольором.

2.1.2 Акумулятор

Абсолютна більшість інструкцій **IL** виконують деяку операцію зі змістом акумулятора (див. його визначення трохи нижче). Операнд, звичайно, теж бере участь в інструкції, але результат знову поміщається в акумулятор. Наприклад, інструкція **SUB 10** віднімає число 10 від значення акумулятора й поміщає результат в акумулятор. Команди порівняння порівнюють значення операнда й акумулятора, результат порівняння **ІСТИНА (TRUE)** або **НЕПРАВДА (FALSE)** знову поміщається в акумулятор. Команди переходу на мітку здатні аналізувати акумулятор і приймати рішення – виконувати перехід чи ні.

Акумулятор **IL** є універсальним контейнером, здатним зберігати значення змінних будь-якого типу.

В акумулятор можна помістити значення типу **BOOL,** потім **INT** або **REAL,** транслятор не буде вважати це помилкою. Така гнучкість не означає, що акумулятор здатний одночасно мати кілька значень різних типів. Тільки одне, причому тип значення також фіксується в акумуляторі. Якщо операція вимагає значення іншого типу, транслятор видасть помилку.

У стандарті МЕК замість терміна «акумулятор» використовується термін «результат» **(result)**. Так, інструкція бере «поточний результат» і формує «новий результат». Проте майже всі посібники із програмування різних фірм широко використовують термін «акумулятор».

2.1.3 Перехід на мітку

Програма на **IL** виконується підряд, зверху вниз. Для зміни порядку виконання й організації циклів застосовується перехід на мітку. Перехід на мітку може бути безумовним **JMP** – виконується завжди, незалежно від будь-чого. Умовний перехід **JMPC** виконується тільки при значенні акумулятора **ІСТИНА**. Перехід можна виконувати як уверх, так і вниз. Мітки є локальними, інакше кажучи, перехід на мітку в іншому POU не допускається.

Переходи потрібно організовувати досить акуратно, щоб не отримати нескінченний цикл:

```
LD      5
ST      Counter
loop1:
(*тіло циклу*)
LD      Counter
ADD     3
ST      Counter
LE      20
JMP     loop1
```

У прикладі показана реалізація циклу на 5 повторень із однією очевидною помилкою. Замість безумовного переходу **JMP** повинен бути **JMPC**.

У системах з інтерпретатором IL або проміжним кодуванням час виконання переходу є залежним від напрямку й відстані. У CoDeSys команда безумовного переходу транслюється в одну машинну команду й виконується дуже швидко. Обмежень на число переходів у CoDeSys немає.

Але це не означає доцільність створення більших монолітних IL-програм з множиною переходів. Такі програми дуже складно читати й нелегко супроводжувати.

2.1.4 Дужки

Послідовний порядок виконання команд **IL** можна змінювати за допомогою дужок. Дужка, що відкривається, ставиться в інструкції після операції. Дужка, що закривається, ставиться в окремому рядку. Інструкції, укладені в дужки, виконуються в першу чергу. Результат обчислення інструкцій у дужках поміщається в додатковий акумулятор, після чого виконується команда, що має відкриваючу дужку. Наприклад:

```
LD      5
MUL     (2
SUB     1
)
```

```
ST      y       (*y=5*(2–1)=5*)
LD      5
MUL     2
SUB     1
ST      y       (*y=5*2–1=9*)
```

Дужки можуть бути вкладеними. Кожне вкладення вимагає організації якогось тимчасового акумулятора. Це викликає неоднозначність при виході із блока дужок командами **JMP**, **RET**, **CAL** і **LD**. Застосовувати ці команди в дужках не можна.

2.1.5 Модифікатори

Додавання до мнемоніки деяких операторів символів – модифікаторів **'C'** і **'N'** модифікує зміст інструкції. В **IL** можна використовувати наступні оператори й модифікатори:

Модифікатори:

C із **JMP**, **CAL**, **RET**: інструкція виконується тільки тоді, коли результат акумулятора **ІСТИНА**.

N із **JMPC**, **CALC**, **RETC**: інструкція виконується тоді, коли результат акумулятора **НЕПРАВДА**.

N в інших випадках: інверсія операнда.

Символ **'N'** (negation) викликає інверсію значення операнда до виконання інструкції. Операнд повинен бути типів **BOOL**, **BYTE**, **WORD** або **DWORD**.

Символ **'C'** (condition) додає перевірку умов до команд переходу, виклику й повернення. Команди **JMPC**, **CALC**, **RETC** будуть виконуватися тільки при значенні акумулятора **ІСТИНА**. Додавання символу **'N'** приводить до порівняння умови з інверсним значенням акумулятора. Команди **JMPCN**, **CALCN**, **RETCN** будуть виконуватися тільки при значенні акумулятора **НЕПРАВДА**. Модифікатор **'N'** без **'C'** не має сенсу в даних операціях і не застосовується.

2.1.6 Оператори

Стандартні оператори **IL** із припустимими модифікаторами надані в таблиці.

Оператор	Модифікатор	Опис
LD	**N**	Завантажити значення операнда в акумулятор
ST	**N**	Присвоїти значення акумулятора операнду
S		Якщо акумулятор **ІСТИНА**, установити логічний операнд (**ІСТИНА**)
R		Якщо акумулятор **ІСТИНА**, установити логічний операнд (**НЕПРАВДА**)
AND	**N, (**	Порозрядне **І**
OR	**N, (**	Порозрядне **АБО**
XOR	**N, (**	Порозрядне виключне **АБО**
NOT		Порозрядна інверсія акумулятора

ADD	**(**	Додавання
SUB	**(**	Віднімання
MUL	**(**	Множення
DIV	**(**	Ділення
MOD	**(**	Ділення за модулем
GT	**(**	>
GE	**(**	> =
QE	**(**	=
NE	**(**	< >
LE	**(**	< =
LT	**(**	<
JMP	**CN**	Перехід до мітки
CAL	**CN**	Виклик функціонального блока
RET	**CN**	Вихід із POU і повернення в програму, яка її викликає

Оператори **S** і **R** застосовуються тільки з операндами типу **BOOL**. Інші оператори працюють із будь-якими змінними базових типів.

Наведений список містять оператори, підтримувані в обов'язковому порядку. Транслятори коду CoDeSys для різних апаратних платформ реалізують різні підмножини додаткових операторів.

Приклад **IL** програми з використанням деяких модифікаторів:

LD TRUE (*завантажити значення **ІСТИНА** в акумулятор*)
ANDN BOOL1 (*виконати **I** с інверсним значенням змінної **BOOL1***)
JMPC mark (*якщо значення акумулятора **ІСТИНА**, то перейти до мітки **"mark"***)
LDN BOOL2 (*зберегти інверсне значення **BOOL2** в акумуляторі*)
ST ERG (*зберегти значення акумулятора в **ERG***)

Після оператора можна поставити дужки, тоді значення виразу всередині дужок розглядається як операнд.

Наприклад:

```
LD      2
MUL     2
ADD     3
ST      ERG
```

Тут значення **ERG** дорівнює 7. Якщо поставити дужки, то порядок обчислень зміниться:

```
LD      2
MUL     (2
ADD     3
)
ST      ERG
```

Тепер значення змінної **ERG** дорівнює 10.

Операція **MUL** виконується тільки тоді, коли програма доходить до **")"**. В якості операнда операція **MUL** використовує значення **5**.

2.1.7 Виклик функціональних блоків і програм

Викликати екземпляр функціонального блока або програму в **IL** можна з одночасним присвоєнням змінних. Наприклад:

```
CAL     CTD_1(CD: - TRUE, LOAD := FALSE, PV := 10)
LD      CTD_1.CV
ST      y
```

Аналогічний виклик можна виконати з попереднім присвоєнням значень вхідних змінних:

```
LD      TRUE
ST      CTD_1.CD
LD      FALSE
ST      CTD_1.LOAD
LD      10
ST      CTD_1.PV
CAL     CTD_1
LD      CTD_1.CV
ST      y
```

2.1.8 Виклик функції

При виклику функції з перерахуванням параметрів в **IL** існує одна немаловажлива особливість. Як перший параметр використовується акумулятор:

```
LD      TRUE
SEL     3,4
```

На **ST** це рівносильна виклику **SEL (TRUE,3,4)**. Очевидно, при виклику функції або оператора з одним параметром список параметрів взагалі не потрібний:

```
LD      ivarl
INT_TO_BOOL
ST      bvarl
```

2.1.9 Коментування тексту

Мова **IL** є мовою низького рівня. Тому тексти **IL** завжди досить великі за обсягом. Звичайно при написанні програми алгоритм здається настільки зрозумілим, що не має потреби в коментарях. Однак текст чужої програми без коментарів зрозуміти дуже важко. Нерідко буває, що й самому автору програми через рік-інший після її написання важко в ній розібратися. Можливість коментувати кожний рядок не означає, що так і потрібно робити. Правильно складене пояснення повинне пояснювати суть проблеми й ідею рішення, а не описувати самі команди. Наприклад: (*Ігнорувати коливання до 5 одиниць*) – поганий коментар. (*Коливання вимірів до 5 одиниць є шумом*) – значно краще.

Транслятор **IL CoDeSys** допускає богаторядкові коментарі. Цілісне пояснення завжди сприймається краще, ніж короткі уривкові коментарі в рядках інструкцій.

Коментарі МЕК мають один важливий недолік. Якщо при налагодженні знадобиться тимчасово відключити частину вихідного тексту, то простіше за все його цілком закоментувати. Тут звичайно й виникає проблема вкладених коментарів. Вкладені коментарі не настільки страшні для транслятора **CoDeSys**, скільки завдають незручності при виправленні тексту. Строкові коментарі (які уводять за допомогою **;** – в асемблері й // – в C++), на жаль, стандартом МЕК не передбачені.

2.1.10 IL у режимі виконання

У режимі виконання **CoDeSys** автоматично перетворює вікно редактора у вікно моніторингу. Значення всіх змінних відображаються праворуч від команд **IL** і доступні для зміни. Є можливість встановити або скинути точку зупину й пройти пошагово програму за одною командою. У режимі **Flow control** (Відображати потік виконання) **CoDeSys** у вікні моніторингу підсвічуються номера рядків, які виконувалися в попередньому робочому циклі, і відображаються відповідні значення акумулятора.

Приклад:

```
  LD      17
  ST      lint (* коментар*)
  GE      5
  JMPC    next
  LD      idword
  EQ      istruct.sdword
  STN     test
next:
```

3 Текстові редактори

3.1 Робота в текстових редакторах

Текстові редактори (використовувані для написання текстів програм **IL** і **ST**) в **CoDeSys** забезпечують звичайні функції текстових редакторів Windows. Текстові редактори підтримують кольорове синтаксичне виділення. Коректно введені інструкції виділяються блакитним кольором.

У режимі заміни напис **OV** у статусному рядку стає чорною. Натискаючи клавішу <**Ins**>, можна перемикатися між режимами вставки й заміни.

Приклад відображення інформації на екран монітора приведений на рис. 3.1.

Найбільш важливі команди знаходяться у контекстному меню, що з'являється при клацанні правої кнопки миші або при натисканні поєднання клавіш <**Ctrl**> +<**F10**>.

У текстових редакторах доступні наступні команди меню:

“Вставка” “Оператор” (“Insert” “Operator”)

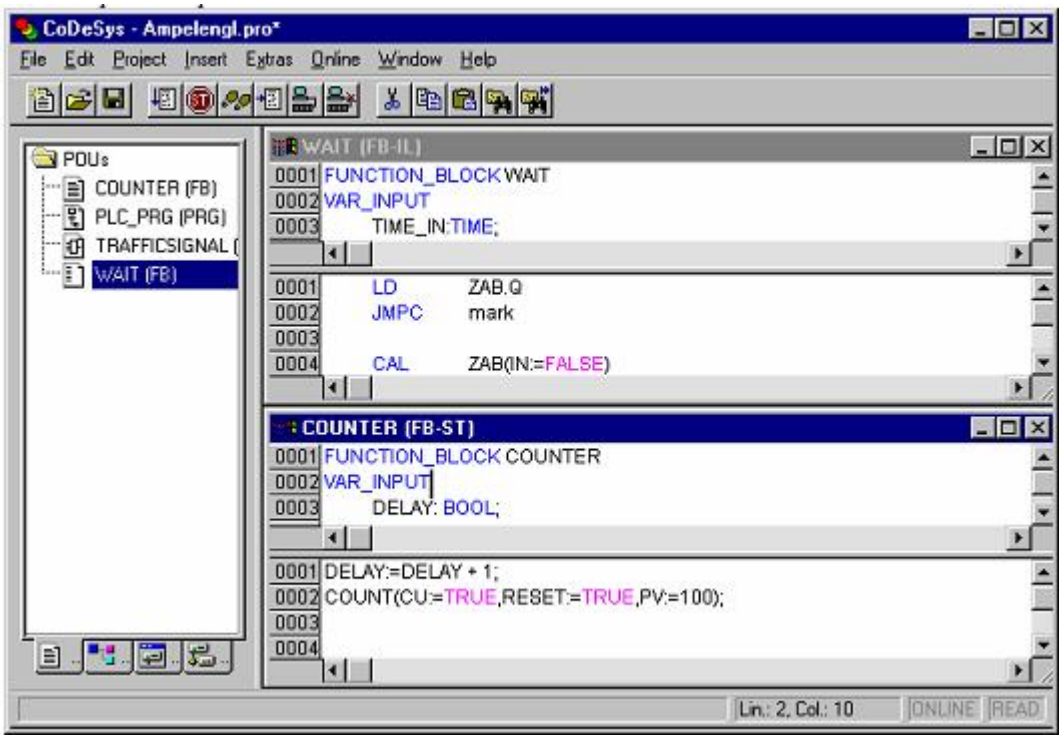

Рисунок 3.1 – Вид в режимі редагування

Викликає список всіх доступних для відповідної мови операторів. Якщо вибрати оператор зі списку й натиснути кнопку OK, то обраний оператор буде доданий у поточну позицію курсора.

“Вставка” “Операнд” (Insert” “Operand”)

Виводить на екран список усіх доступних змінних. Можна вибрати категорію змінних (глобальні, локальні, системні), які будуть зображені у списку.

Якщо операнд обраний і натиснута кнопка OK, то обраний операнд буде вставлений у поточну позицію курсора (аналогічно роботі Input Assistant).

“Вставка” “Функція” (“Insert” “Function”)

Виводить діалогове вікно, у якому можна вибрати функцію зі списку стандартних або певних користувачем функцій.

Обрана функція поміщається в поточну позицію курсора після натискання кнопки OK.

Якщо обрано прапор **“З Аргументом” (“With Argument”)**, то також будуть вставлені необхідні вхідні й вихідні змінні.

“Вставка” “Функціональний Блок” (“Insert” “Function Block”)

Виводить список усіх доступних у проекті функціональних блоків. Можливо вибрати, які функціональні блоки будуть відображені: або стандартні, або визначені користувачем.

Обраний функціональний блок поміщається в поточну позицію курсора при натисканні клавіші OK.

Якщо обрано прапор **“With Arguments”**, то з'являться необхідні вхідні й вихідні змінні.

Виклик POU з вихідними параметрами

У текстових мовах **ST** і **IL** вихідні параметри **POU** можна зв'язати з будь-якими змінними прямо при виклику **POU**.

Приклад: Вихідний параметр **out1** присвоюється змінній ***a***.

IL: CAL afbinst(in1:=1, out1=>*a*)

ST: afbinst(in1:=1, out1=>*a*);

Якщо **POU** уводиться за допомогою **Асистента введення** (<**F2**>) з опцією **"With arguments"**, то виклик у **ST** або **IL** автоматично відображається з таким синтаксисом для всіх параметрів. Однак не обов'язково всі їх використовувати.

3.2 Текстові редактори в режимі Online

Текстові редактори **CoDeSys** поєднують типові функції сучасних налагоджувачів. У текстових редакторах підтримуються такі **Online**-функції, як установка точок зупину й виконання програми за кроками.

У режимі **Online** вікно текстового редактора розділяється по вертикалі на дві частини. У лівій частині вікна розташований текст програми, а в правій значення змінних. Ширину частин можна змінювати, перетаскуючи мишкою межу між ними.

Перегляд значень змінних здійснюється так само, як і в редакторі розділу оголошень.

Коли зв'язок з контролером установлений, на екран виводяться поточні значення змінних.

При моніторингу виразів виводиться підсумкове значення. Наприклад: ***a*** **AND** ***b*** відображається з рядком **":=TRUE"**, якщо ***a*** і ***b*** істинні.

Для біт адресованих змінних виводиться значення відповідного біта (наприклад, *a*.2 зображується з рядком **":=TRUE"**, якщо ***a*** має значення **4**).

Якщо помістити покажчик миші на змінну, то у підказці, яка спливає, буде виведений коментар, тип і адреса змінної.

"Доповнення" "Опції Моніторингу" ("Extras" "Monitoring Options")

Ця команда дозволяє змінити настроювання вікна, у якому ви переглядаєте значення змінних. У текстових редакторах під час моніторингу вікно розділяється на дві частини. Текст програми перебуває в лівій частині, а змінні, що переглядаються, в правій частині вікна.

Можна установити ширину **Width** вікна монітора й інтервал **Distance** між двома змінними в рядку. Значення інтервалу, що дорівнює 1, відповідає висоті обраного шрифту.

Зверніть увагу, що ширину вікон можна оперативно змінювати, перетягуючи мишкою межу між ними.

Діалог установки опцій вікна моніторингу (рис. 3.2).

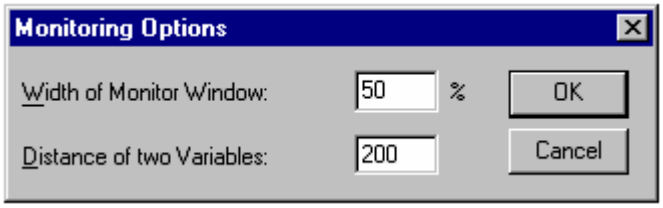

Рисунок 3.2 – Діалог установки опцій вікна моніторингу

3.3 Точки зупину

Тому що у **CoDeSys** кілька рядків на **IL** поєднуються при компіляції, то точки зупину не можна установлювати в довільному рядку. Точки зупину установлюються там, де можуть змінитися або значення змінних, або напрям виконання програми. Виключення становлять точки виклику функції. Тут також можна поставити точку зупину. У позиціях, що перебувають між перерахованими вище, точка зупину не мала б смислу, тому що тут не змінюються ні дані, ні напрямок виконання програми.

У мові **IL** точки зупину можна ставити в наступних позиціях:

На початку кожного **POU**

На кожному операторі **LD, LDN**

На кожному операторі **JMP, JMPC, JMPCN**

На кожній мітці

На кожному операторі **CAL, CALC, CALCN**

На кожному операторі **RET, RETC, RETCN**

Наприкінці кожного **POU**

Мова **ST** допускає наступні позиції точок зупину:

На кожній інструкції присвоювання

На будь-якій інструкції **RETURN** і **EXIT**

У позиціях, де обчислюються умови **(WHILE, IF,REPEAT)**

Наприкінці **POU**

При установці точок зупину, номер відповідного рядка виділяється кольором, обраним в опціях проекту.

Редактор **IL** з припустимими позиціями точок зупину (номери таких рядків виділені темно-сірим) (рис. 3.3).

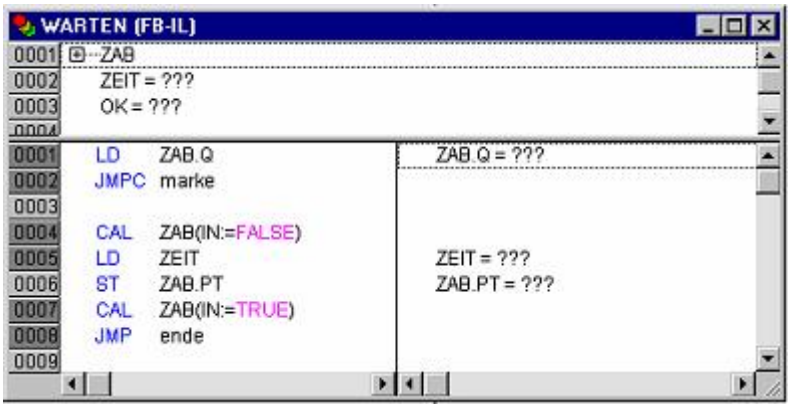

Рисунок 3.3 – Редактор **IL** с припустимими позиціями точок зупину

Як поставити точку зупину?

Для того щоб поставити точку зупину, клацніть мишкою по номеру рядка, у якому ви хочете поставити точку зупину. Колір номера рядка зміниться з темно-сірого на блакитний, і точка зупину буде встановлена в **PLC**.

Видалення точок зупину

Для цього клацніть по номеру рядка, у якому установлена точка зупину. Установлювати й видаляти точки зупину також можна через меню **“Онлайн“**

“Точка зупину“ (“Online“ “Toggle Breakpoint”), натискаючи кнопку <**F9**> або кнопку на панелі інструментів.

Що відбувається в точках зупину

Коли точка зупину буде досягнута, номер виділеного рядка стане червоним. Програма буде зупинена в **PLC**.

Якщо програма зупинена, то її виконання можна продовжити командою **“Онлайн” “Старт” (“Online” “Run”).**

Крім того, ви можете скористатися командами **“Онлайн” “Крок вверх”** і **“Крок вниз” (“Online” “Step over”** і **“Step in”)** для виконання програми по кроках. Якщо користуватися командою **“Step over”**, програма не буде зупинятися в точках, викликуваних **POU**. При виклику команди **“Step in”** по кроках проходяться всі виклики **POU**.

Номер рядка в текстовому редакторі

Номер рядка в текстовому редакторі визначає номер рядка тексту **POU**.

У режимі **“Оффлайн” (“Offline”)** клацання по певному номеру рядка приводить до виділення текстового рядка.

У режимі **“Онлайн” “Online”** колір номера рядка визначає, чи установлена точка зупину в цьому рядку чи ні.

Ось стандартні установки для кожного кольору:

темно-сірий: рядок, у якому можна установити точку зупину;

блакитний: точка зупину установлена в цьому рядку;

червоний: програма зупинена в цій точці.

Редактор мови IL

Ось так виглядає програма, написана на **IL** у відповідному редакторі **CoDeSys** (рис. 3.4).

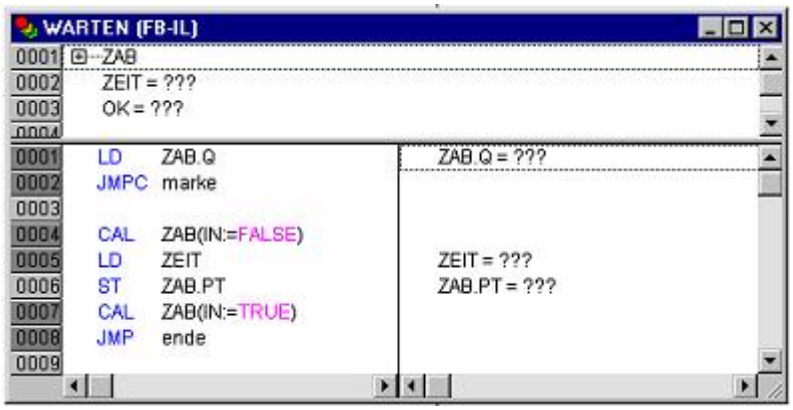

Рисунок 3.4 – Програма, яка написана на **IL** у відповідному редакторі

Всі редактори **POU** складаються з розділу оголошень і властне тіла програми, відділених один від одного роздільником.

Редактор **IL** – це текстовий редактор зі звичайними функціями текстового редактора **Windows**.

В **CoDeSys** допустимий виклик **POU** з декількома вкладеними викликами:

Приклад:

```
CAL CTU_inst(
    CU:=%IX10,
    PV:=(
    LD A
    ADD 5
        )
    )
```

Інформацію, що стосується мови програмування, можна знайти у розд. 2.2.1 Instruction List (IL) [5].

IL у режимі Online

Якщо викликати команду **“Онлайн” “Відображати потік виконання” (“Online” “Display Flow control”)**, у лівій частині кожного рядка з'явиться поле, що містить значення акумулятора. Більш докладна інформація про редактор **IL** у режимі **Online** описана у розділі “Текстові редактори в режимі Online” [5].

4 Контрольні питання

4.1 Пояснити різницю між глобальною й локальною змінною.
4.2 Як можна змінювати значення змінних у режимі Онлайн?
4.3 Як програмуються розгалуження й цикли мовою **IL**?
4.4 З яких полів складається інструкція мовою **IL**?
4.5 Синтаксис коментарів мовою **IL**?
4.6 Які відомі рекомендації при виборі імен змінних у програмі?
4.7 Які змінні важливо розміщувати в розділі оголошень?
4.8 Які відомі модифікатори в мові **IL** і що вони виконують?

5 Домашнє завдання

5.1 Виконати розробку алгоритму керування ділянки заданого технологічного процесу у вигляді словесного опису й блок схеми для його подальшого програмування.

5.2 Підготувати текст програми мовою **IL** з коментарями.

6 Лабораторне завдання

6.1 Запустити середовище розробки **CoDeSys**. Якщо в організаторі об’єктів автоматично відкривається проект, потрібно закрити його за допомогою меню **Файл → Закрити (File → Close)** [2, 3].

6.2 Відкрити новий проект за допомогою кнопки **New (Створити)** на панелі інструментів або меню **Файл → Відкрити (File → Open)**. У вікні, що відкрилося, **Настроювання цільової платформи (Target Setting)** виберіть **None.** У новому вікні, що відкрилося, **Новий програмний компонент (New**

POU) вибрати тип **POU «Програма»** і мова програмування **ST**. Натисніть кнопку ОК.

6.3 В робочій області, що з'явилася, за допомогою клавіатури ввести програму, яка виконує алгоритм логічного керування ділянки технологічного процесу згідно із домашнім завданням. Додати коментарі для окремих частин програми. При написанні програми слід зазначати змінні, які характеризують змістовне значення параметра і правильно вибрати найменування та тип змінних. Зберегти проект під унікальним ім'ям. Занести до протоколу перелік програм **(PRG)**, які входять у даний проект. Цей перелік відображається в організаторі об'єктів.

6.4 Ввімкнути режим симуляції за допомогою меню **Онлайн → Режим сімуляції (Online → Simulation Mode)**.

6.5 Під'єднати середовище до симуляції ПЛК за допомогою меню **Онлайн → Підключення (Online → Login)**. Звернути увагу на те, що під час під'єднання виконується компіляція проекту.

6.6 Запустити проект на виконання за допомогою меню **Онлайн → Старт (Online → Run)** та виконати його тестування на правильність виконання.

6.7 Після тестування записати його в зошиті для лабораторних робіт.

7 Зміст протоколу

Протокол лабораторної роботи "Програмування ПЛК. Вивчення мови IL" оформлюється в зошиті для лабораторних робіт. Протокол повинен мати назву лабораторної роботи та її мету, відповіді на контрольні питання, хід виконання домашнього та лабораторного завдання, блок-схеми алгоритмів програм та коментарі до програм розроблених та налагоджених під час виконання лабораторної роботи, висновки.

8 Перелік рекомендованої літератури

1. Петров И.В Программируемые контроллеры. Стандартные языки и приемы прикладного программирования / Под ред. проф. В.П. Дьяконова – М: ООО «СОЛОН-Пресс», 2004. – 256 с.

2. Методичні посібники для лабораторних робіт даного курсу.

3. Конспект лекцій з курсу МП і ПЗА.

4. Веб-сторінка фірми Smart Software Solutions GmbH виробника середовища CoDeSys http://www.3s-software.com/

5. Веб-сторінка ПК "Пролог", підтримка середовища CoDeSys на російській мові http://www.codesys.ru/, технічна документація CoDeSys_V_23_RU.

6. Христенсен, Дж. Х. Знакомство со стандартом на языки программирования PLC: IEC 1131-3 (МЭК 1131-3) [Электронный ресурс]/ Дж. Х. Христенсен. – [2004]. – Режим доступа: http://www.asutp.ru

7. Руководство пользователя по программированию ПЛК в CoDeSys V2.3. – Смоленск: ПК "Пролог", 2004. – 423 с.

8. Деменков Н.П. Языки программирования промышленных контроллеров: Учебное пособие / Под ред. К.А. Пупкова.– М.: Изд-во МГТУ им. Н.Э. Баумана. – 172 с.