

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДОНБАСЬКА ДЕРЖАВНА МАШИНОБУДІВНА АКАДЕМІЯ

МЕТОДИЧНІ ВКАЗІВКИ

до комп'ютерного практикуму

по дисципліні

«ПРОЕКТУВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ»

Частина 2

(для студентів спеціальності 151

“Автоматизація та комп'ютерно-інтегровані технології»)

Краматорськ 2018

Методичні вказівки до комп'ютерного практикуму по дисципліні "Проектування систем автоматизації". Частина 2. (для студентів спеціальності 151 "Автоматизація та комп'ютерно-інтегровані технології») / Укл.. О. О. Сердюк. - Краматорськ: ДДМА, 2018 – 91 с.

Освітлені принципи роботи в графічній оболонці Simatic Manager інструментальної системи STEP 7, правила й приймання конфігурування апаратури центральних станцій SIMATIC S7-300/400, а також децентралізованої периферії систем автоматизації на шині PROFIBUS. Викладені методики створення програм у редакторі LAD/FBD/STL, налагодження програм у додатку S7-PLCSIM, а також методика програмування систем автоматизації мовою S7-HiGraph. З використанням прикладів показана методика створення графа стану, координуючого графа, а також групового графа, який підтримує діагностичні й алармові функції.

Наведені варіанти індивідуальних завдань для самостійної роботи.

Укладач

О. О. Сердюк, доц.,

Відп. за випуск

Г. П. Клименко, проф..

ЗМІСТ

ВСТУП.....	4
1 КОНФІГУРУВАННЯ Й ПАРАМЕТРУВАННЯ АПАРАТУРИ ЦЕНТРАЛЬНИХ СТАНЦІЙ У СЕРЕДОВИЩІ STEP 7	5
1.1 Порядок конфігурування й параметрування стійок	5
1.2 Методичні вказівки по конфігуруванню станцій S7-300/400	8
1.3 Методика параметрування модулів і інтерфейсів	13
1.4 Методика виконання індивідуального завдання	16
2 КОНФІГУРУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ ПЕРИФЕРІЇ У МЕРЕЖІ PROFIBUS	18
2.1 Правила конфігурування децентралізованої периферії.....	18
2.2 Методика створення й параметрування майстер-системи	19
2.3 Конфігурування станцій ET 200	23
2.4 Конфігурування інтелектуальних ведених DP	27
2.5 Методика виконання індивідуального завдання	32
3 ПРОГРАМУВАННЯ ЛОГІЧНОГО ЗАВДАННЯ КЕРУВАННЯ.....	33
3.1 Інтерфейс редактора LAD/STL/FBD	33
3.2 Методика створення проекту програми	34
3.3 Приклад програмування системи керування	37
3.4 Методика виконання індивідуального завдання	46
4 НАЛАГОДЖЕННЯ ПРОГРАМИ В S7-PLCSIM	47
4.1 Загальні відомості про S7-PLCSIM	47
4.2 Методика роботи в S7-PLCSIM	48
4.3 Методика виконання роботи й зміст звіту	52
5 СТВОРЕННЯ ПРОГРАМИ МОВОЮ S7-HIGRAPH	54
5.1 Принцип програмування мовою S7-HiGraph	54
5.2 Приклад виділення графів станів у завданні керування.....	56
5.3 Послідовність створення графа станів в HiGraph.....	60
5.4 Вимоги до звіту по роботі	68
6 РОЗРОБКА Й НАЛАГОДЖЕННЯ ПРОГРАМИ HIGRAPH.....	70
6.1 Створення групового графа	70
6.2 Компіляція файлів програми	76
6.3 Завантаження програми в контролер та її налагодження.....	79
6.4 Порядок виконання роботи й вимоги до звіту	81
Додаток А. Варіанти індивідуальних завдань до роботи 1	82
Додаток Б. Варіанти індивідуальних завдань до роботи 2.....	84
Додаток В. Варіанти індивідуальних завдань до робіт 3 і 4	86
Додаток Г. Базові функції STL	87

ВСТУП

Розробка системи автоматизації являє собою процес поетапного проектування апаратури й програмного забезпечення. Створення проекту здійснюється в графічній оболонці середовища STEP 7 – SIMATIC Manager.

На першому етапі цього процесу визначається склад апаратного забезпечення системи і у додатках Hardware Configuration і Netpro проводиться конфігурування центральної станції й периферійних модулів. Кінцевим результатом цього етапу є створення файлу конфігурації системи.

Процес, який повинен бути автоматизований, при більш детальному розгляді розділяється на ряд приватних завдань, з'єднаних між собою. Кожне завдання вирішується в рамках певних апаратних і програмних ресурсів.

Розробка програм може проводитися в декількох редакторах, кожний з яких дозволяє створювати програми з різним рівнем вистави. Створення програм логічного рівня з використанням булевої алгебри забезпечує редактор LAD/STL/FBD.

Незважаючи на те, що всі редактори здійснюють перевірку правильності синтаксису використовуваної мови, у програмі можливі помилки, пов'язані з логікою роботи програми. У зв'язку із цим необхідно робити тестування й налагодження програми. Для виконання цієї роботи використовується додаток S7-PLCSIM, за допомогою якого можна симулювати реальний контролер і перевірити якість функціонування програми на віртуальному контролері. Використання S7-PLCSIM дозволяє усунути помилки програмування й переконатися в працездатності програми.

Програмна система STEP 7 призначена для виконання всього комплексу проектування – конфігурування, програмування й тестування системи автоматизації.

Важливість робіт з конфігурування визначається тим, що програмне забезпечення неможливе буде реалізувати в апаратурі без взаємного узгодження всіх ресурсів системи, а також прив'язки конкретних вхідних і вихідних сигналів (змінних користувачької програми) до конкретних ліній зв'язку, вузлів мережі й модулів станцій.

Цей практикум призначений для освоєння методики й правил виконання робіт з конфігурування, програмування й налагодженню систем автоматизації SIMATIC.

1 КОНФІГУРУВАННЯ Й ПАРАМЕТРУВАННЯ АПАРАТУРИ ЦЕНТРАЛЬНИХ СТАНЦІЙ У СЕРЕДОВИЩІ STEP 7

Ціль роботи: освоєння інтерфейсу інструментальної системи STEP 7, а також методики конфігурування й параметрування апаратури центральних станцій SIMATIC S7.

1.1 Порядок конфігурування й параметрування стійок

Конфігурування

Під *конфігуруванням* розуміється розміщення інтерфейсних, функціональних і комунікаційних модулів, а також стійок у вікні станції. Стійки представляються за допомогою конфігураційної таблиці, яка, як і реальна стійка, допускає певне число встановлюваних модулів.

Параметрування

Під *параметруванням* розуміється установка властивостей модулів. Наприклад, для CPU встановлюється час контролю циклу, для шини PROFIBUS встановлюються параметри шини, параметри провідних та ведених модулів. Параметризація дозволяє легко замінити модулі, тому що встановлені параметри автоматично завантажуються в новий модуль у процесі запуску.

Коли потрібно конфігурування апаратури?

Властивості програмувальних контролерів і модулів S7 встановлюються за замовчуванням. Однак в наступних випадках конфігурування обов'язкове:

- якщо необхідно змінити параметри модуля, встановлені за замовчуванням, наприклад, дозволити для модуля переривання від процесу;
- якщо потрібно проектувати комунікаційні з'єднання;
- якщо використовується шина PROFIBUS-DP, на яку встановлюються станції з децентралізованою периферією;
- якщо створюються станції S7-400 з декількома CPU або стійками розширення;
- якщо проектують системи підвищеної надійності (H-системи).

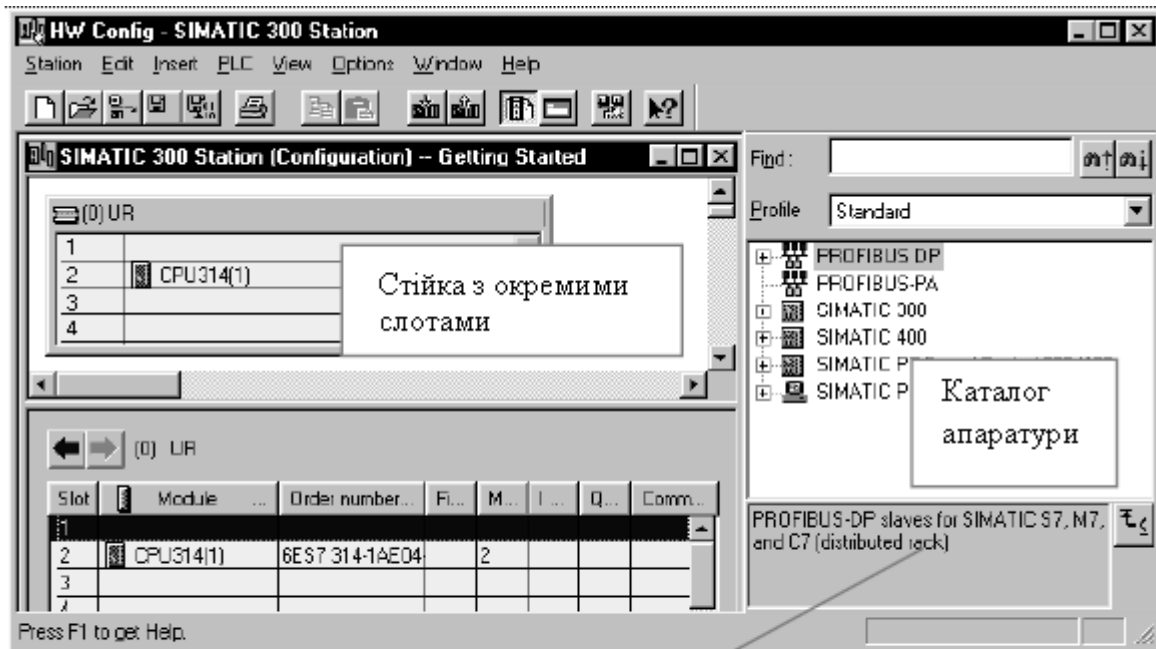
Основний порядок конфігурування апаратури

Для конфігурування системи автоматизації в додатку Hardware Configuration використовуються два вікна:

- вікно станції SIMATIC Station, у якому розміщаються стійки;
- вікно Hardware Catalog (*Каталог апаратури*), з якого вибираються необхідні апаратні компоненти (стійки, сигнальні й інтерфейсні модулі).

На рисунку 1.1 центральна стійка позначена (0)UR, де 0 – порядковий номер стійки, UR – тип стійки (Universal Rack – універсальна стійка).

Процес конфігурування полягає в тому, що необхідні компоненти вибираються у вікні Hardware Catalog і переносяться у вікно станції.



Конфігураційна таблиця

Коротка інформація по вибраному елементу

Рисунок 1.1 – Розташування вікон інтерфейсу в середовищі конфігурування HW Config

Центральна станція складається з головної стійки й стійок розширення. Компонування станції відображається в конфігураційній таблиці стійки, розташованій під вікном станції (рис. 1.2). У таблиці відображаються номери слотів, найменування модулів, їх адреси й замовні номери.

Слот	Модуль	Номер для заказа	Адрес MPI	Комментарий
Slot	Module	Order number	Firmware MPI address	I addr... Q addr... Comment
1	PS 307 10A	6ES7 307-1KA00-0AA0		
2	CPU 314	6ES7 314-1AE01-0AB0	2	
3				
4	D18xAC120/230V	6ES7 321-1FF10-0AA0		0
5	A18x12Bit	6ES7 331-7KF02-0AB0		272...287
6	A18x16Bit	6ES7 331-7NF10-0AB0		288...303
7	A18xTC/4xRTD, Ex	6ES7 331-7SF00-0AB0		304...319
8	AQ2x12Bit	6ES7 332-5HB00-0AB0		320...323
9	AQ2x12Bit	6ES7 332-5HB81-0AB0		336...339
10				
11				

Рисунок 1.2 – Вид таблиці конфігурування стійки

Основний порядок параметрування

Після того, як компонент розміщений у вікні станції, можна перейти в режим діалогу для зміни встановлених за замовчуванням параметрів або адрес (режим параметрування).

Для переходу у вікно установки властивостей компонента можна застосувати один зі способів:

- двічі клацнути на компоненті лівою кнопкою миші;
- вибрати команду меню Edit ⇒ Object Properties (*Редагування* ⇒ *Властивості об'єкта*).
- за допомогою правої кнопки миші вибрати зі спливаючого меню команду Object Properties (*Властивості об'єкта*).

Для настроювання поведінки системи особливе значення мають властивості CPU. На закладках CPU можна встановити характеристики запуску, області локальних даних і пріоритети для переривань, області пам'яті, характеристики реманентності (збереження даних у пам'яті після вимикання живлення), тактові меркери, рівень захисту й пароль.

У закладці General (*Загальне*) можна параметрувати інтерфейси, наприклад, MPI або вбудований інтерфейс PROFIBUS-DP.

Що слід знати про правила, що ставляться до слотів?

STEP 7 контролює правильність конфігуруванні станції. При цьому автоматично перевіряються адресні області, так що та сама адреса не може бути зайнята двічі.

Порядок розташування модулів у слотах показано на рисунку 1.3.

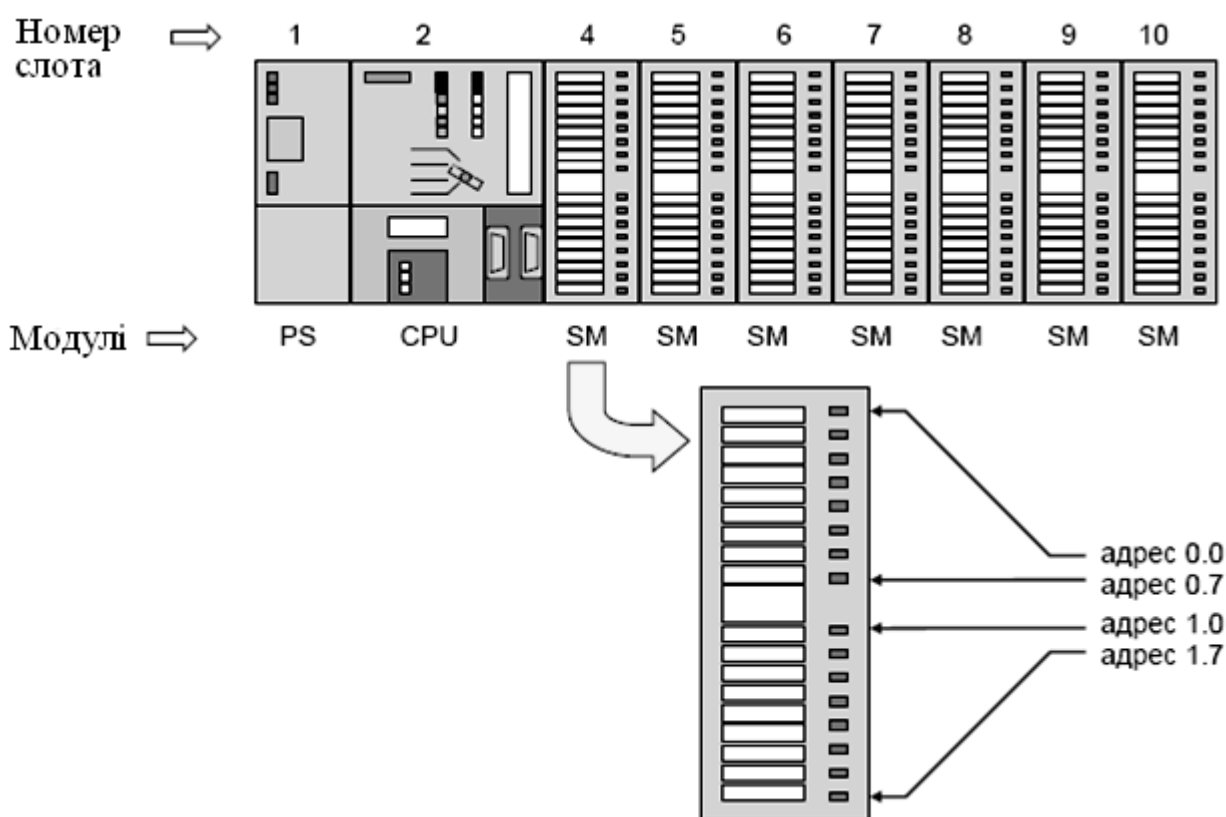


Рисунок 1.3 – Схема розташування модулів у слотах і автоматичної адресації входів-виходів

Слот 1 призначений для установки модуля живлення.

Процесорний модуль повинен бути встановлений в слот 2 головної станції.

У станціях S7-300 слот 3 резервується для установки інтерфейсного модуля, а слот 4 є першим настановним місцем для установки сигнальних модулів (SM), комунікаційних процесорів (CP), функціональних модулів (FM).

На рисунку 1.3 показане розміщення модулів. Сигнальний модуль встановлений в слоті 4. Саме з цього слота почнеться адресація входів-виходів. Для кожного слота зарезервовано 4 байта адреси. Тоді неважко визначити, що в цифровому модулі, наприклад, слота 6 адресація буде починатися з 8.0.

Слід мати на увазі, що адреси входів позначаються символом I, наприклад, I 4.0, а адреси виходів – символом Q.

Збереження конфігурації

Щоб зберегти конфігурацію з усіма встановленими параметрами й адресами, необхідно вибрати команду меню Station ⇒ Save and Compile (*Станція ⇒ Зберегти й компілювати*). Для збереження незакінченої конфігурації виберіть команду меню Station ⇒ Save.

1.2 Методичні вказівки по конфігуруванню станцій S7-300/400

Створення станції

Станція може створюватися тільки безпосередньо під проектом. Тому спочатку необхідно виділити проект у лівій частині вікна, а потім вибрати команду меню Insert ⇒ Station ⇒ SIMATIC 300-station, (*Вставити ⇒ Станція ⇒ Станція SIMATIC 300*) або ... (SIMATIC 400-station.

Станція створюється з іменем, даним за замовчуванням. Це ім'я можна замінити іншим, більш інформативним.

Після створення станції необхідно:

1. Виділити у вікні проектів об'єкт Station, після чого в правій частині вікна станції стає видимим об'єкт Hardware (*Апарат*).



– об'єкт "Station",



– об'єкт "Hardware".

Можна також виділити об'єкт Station і вибрати команду меню Edit ⇒ Open Object (*Редагувати ⇒ Відкрити об'єкт*). У результаті на екрані з'являються вікно станції й каталог модулів.

У вікні станції можна помістити стійку та інші компоненти у відповідності зі структурою станції, а з каталогу модулів у вікні Hardware Catalog вибрати необхідні для побудови станції компоненти.

Командою меню Station ⇒ New (*Станція ⇒ Нова*) можна сконфігурувати у тому ж проекті ще одну станцію.

Проектування центральної стійки

Вставка центральної стійки:

1. У вікні Hardware Catalog потрібно вибрати центральну стійку (Rack). Для SIMATIC 300 це профільна шина (Rail), для SIMATIC 400 може бути, наприклад, універсальна стійка (UR1).

2. Використовуючи метод Drag&Drop, слід відбуксирувати стійку у вікно станції. Стійка з'являється у вигляді невеликої конфігураційної таблиці у верхній частині вікна станції. У нижній частині вікна станції з'являється докладна вистава стійки з додатковими даними – замовленим номером, адресою MPI, адресами входів/виходів.

Компонування стійки здійснюється в наступній послідовності:

1. Вибирається модуль із вікна Hardware Catalog. При цьому слоти, у які можна встановити цей модуль, виділяються кольором.

2. З використанням Drag&Drop модуль буксирується у відповідний рядок стійки. При цьому STEP 7 перевіряє, чи не порушені правила для слотів.

3. Кроки 1 і 2 повторюються доти, поки стійка не буде повністю оснащена бажаними модулями.

Примітка: При виділенні слота в стійці можна побачити список усіх можливих для установки модулів. Для цього необхідно правою кнопкою миші відкрити контекстно-залежне меню й у ньому вибрати Insert Object (*Вставити об'єкт*) або Replace Object (*Замінити об'єкт*). Ця можливість рятує від необхідності пошуку апаратури в каталозі.

Відображення інтерфейсів і інтерфейсних модулів

Інтерфейси або інтерфейсні модулі відображаються в конфігураційній таблиці у власному рядку. Цей рядок позначений так само, як і коннектор інтерфейсу, наприклад, XI.

При наявності *вбудованих* інтерфейсів ім'я інтерфейсу з'являється в стовпці Module (*Модуль*). Для установки інтерфейсних модулів потрібно перенести підходящий інтерфейсний модуль (IF) з вікна Hardware Catalog у відповідний рядок, використовуючи Drag&Drop.

Якщо CPU має більш однієї версії операційної системи, то він показується у вікні Hardware Catalog як папка з іконками, що мають різні порядкові номери.

Конфігурування стійок розширення для SIMATIC 300

Для станцій SIMATIC 300 як у якості центральної стійки, так і в якості стійок розширення використовуються тільки профільні рейка (Rail). Кількість профільних рейок визначається реальною конструкцією, однак не повинна бути більше чотирьох.

Стійки розширення з'єднуються в STEP 7 шляхом установки відповідних *інтерфейсних модулів*:

- Для розширення тільки на одну стійку в стійках 0 і 1 встановлюються модулі ІМ 365.
- Для підключення до трьох стійок розширення в стійку 0 встановлюється модуль ІМ 360, а в стійки з 1 по 3 модулі ІМ 361.

Правила заповнення слотів станції S7-300 у стійках полягають у наступному:

Стойка 0:

- Слот 1: Тільки блок живлення, наприклад, 6ES7 307-..., або порожній.
- Слот 2: Тільки CPU (наприклад, 6ES7 314-...).
- Слот 3: Інтерфейсний модуль (наприклад, 6ES7 360) або порожній.
- Слоти з 4 по 11: Сигнальні або функціональні модулі, комунікаційні процесори або порожні.

Стойки з 1 по 3:

- Слот 1: Тільки блок живлення (наприклад, 6ES7 307-...) або порожній.
- Слот 2: Порожній.
- Слот 3: Інтерфейсний модуль.
- Слоти з 4 по 11: Сигнальні або функціональні модулі, комунікаційні процесори (залежно від використовуваного інтерфейсного модуля) або порожні.

Примітка. Порожній модуль (DM 370 Dinttu) – це модуль, який можна встановити замість передбачуваного (тип модуля буде відомий у майбутньому). Порожній модуль дозволяє резервувати адресний простір для майбутнього модуля.

Правила розміщення модулів у станції SIMATIC-400

Правила розміщення модулів у станції S7-400 залежать від типу застосовуваної стійки. У центральній стійці модулі розміщуються за наступними правилами:

- блоки живлення встановлюються тільки в слот 1 і наступні слоти;
- кількість інтерфейсних модулів повинна бути не більше 6, з них не більше 2-х з передачею живлення;
- до центральної стійки через інтерфейсні модулі можна підключити не більше 21 стійки розширення;
- до інтерфейсу *передавального* ІМ 460-1 можна підключити не більш 1 стійки розширення з *передачею струму*;
- до інтерфейсу *передавального* ІМ 460-0 або ІМ 460-3 можна підключити не більш 4 стійок розширення *без передачі струму*.

У стійках можуть встановлюватися резервовані блоки живлення (станції S7-400), для яких слід ураховувати наступні правила:

- установка резервованих блоків живлення можлива тільки в призначені для цього стійки (розпізнаються по більшому замовному номеру й по інформаційному тексту у вікні каталогу апаратури Hardware Catalog);
- резервовані блоки живлення можуть експлуатуватися тільки разом із призначеними для цього CPU (непридатні CPU, наприклад, старої версії, при конфігуруванні відкидаються);
- резервовані блоки живлення повинні встановлюватися зі слота 1 без пропусків;
- резервовані й нерезервовані блоки живлення не можуть вставлятися в одну стійку, тобто змішана експлуатація неможлива.

Конфігурування стійок розширення для S7-400

1. Виберіть підходящу стійку розширення з каталогу апаратури Hardware Catalog.
2. Відбуксируйте стійку, використовуючи Drag&Drop, у вікно станції.
3. Розмістіть в стійці модулі. При цьому блок живлення повинен бути встановлений у першому слоту, а інтерфейсний модуль – *в останньому слоті*.
4. Зробіть з'єднання між інтерфейсними модулями, установленими в центральній стійці й стійці розширення. Із цією метою клацніть двічі на передавальному ІМ. У закладці Connect відображаються всі стійки із установленими ІМ. Виділивши стійку розширення, підключіть її за допомогою екранної кнопки Connect до інтерфейсу передавального ІМ. Підтвердіть з'єднання кнопкою ОК.

Після виконання цих дій між інтерфейсними модулями з'явиться сполучна лінія.

На рисунку 1.4 як приклад показане вікно станції з відображенням центральної стійки (0) і стійки розширення (1), з'єднаних між собою інтерфейсними модулями ІМ 460-0 і ІМ 461-0.

Заміна стійок у станції

Заміна стійок станції SIMATIC S7-400 виправдана, якщо в результаті цієї заміни функціональні можливості станції розширюються. Це відбувається в наступних випадках:

- Заміна стійки, що не підтримує резервування системи живлення, на стійку, що підтримує резервування.
- Заміна короткої стійки (9 слотів) на довгу (18 слотів) для установки додаткових модулів. Для стійок, сконфігурованих як стійки розширення (UR або ER з приймальним ІМ), ІМ автоматично переміщуються в останній слот.
- У станції, спроектованої спочатку з довгою стійкою, стійка може бути замінена короткою. Однак ця заміна неможлива для довгих стійок, спроектованих як стійки розширення.

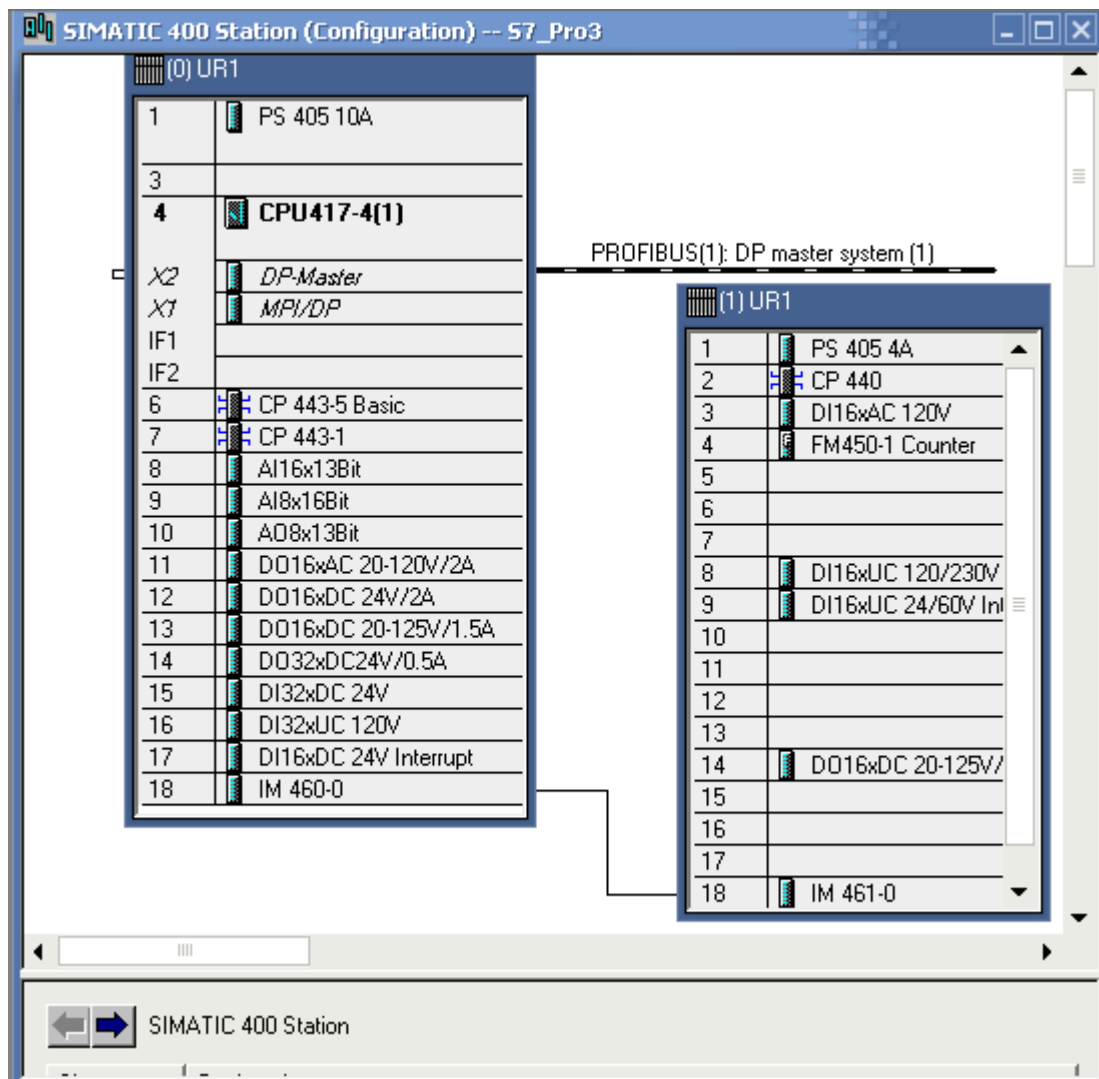


Рисунок 1.4 – Приклад конфігурування стійок

Правила заміни стійок

Стійка станції SIMATIC 400 може бути замінена на іншу, тільки при дотриманні наступних *основних правил* (якщо хоча б одне правило не виконуються, STEP 7 не дозволяє заміну й перериває процедуру з повідомленням про помилку):

- Сегментована стійка (CR2) не може бути замінена несегментованою (наприклад, UR1) і навпаки. Слоти з модулями із двох сегментів не можуть бути однозначно зіставлені слотам в іншій, не сегментованій стійці. Тому стійка CR2 може бути замінена тільки на стійку CR2 з іншим порядковим номером, наприклад, щоб забезпечити установку модулів з резервуванням живлення без того, щоб проводити все конфігурування знову.

- При установці модулів у нову стійку не повинні порушуватися ніякі правила слотів.

- Не дозволяється заміна стійки UR1 із установленим CPU на стійку розширення ER1, тому що установка CPU в ER1 заборонена правилами слотів.

Якщо станція має складну структуру, наприклад, містить кілька стійок, то можна настроїти станцію на мінімальний розмір.

Для мінімізації розміру станції необхідно виконати наступне:

1. Виділити конфігураційну таблицю.
2. Натиснути праву кнопку миші й вибрати в спливаючому меню команду Minimize (*Мінімальний розмір*).

Заміна компонентів

Якщо потрібно замінити компонент у стійці із уже встановленими модулями, виконайте наступне:

1. Виділіть в конфігурації станції компонент, який потрібно замінити.
2. Виділіть у вікні каталогу апаратури ідентичний компонент, сумісний із замінним. Наприклад, для ведених DP можна виділити інтерфейсний модуль IM 153-2.
3. Двічі клацніть на необхідному компоненті в каталозі апаратури. Якщо компоненти сумісні, вони замінюються, причому модулі з вихідної конфігурації зберігають настроювання адрес і параметрів.

Заміна компонента можлива також з використанням буксирування методом drag-and-drop з каталогу апаратури.

Якщо з'являється повідомлення The slot is already occupied (*Слот уже зайнятий*), потрібно спочатку активізувати функцію заміни командою меню Options ⇒ Customize (*Можливості* ⇒ *Настроювання*), а потім вибрати настроювання Enable Module Exchange (*Дозволити заміну модулів*).

1.3 Методика параметрування модулів і інтерфейсів

Параметрування модулів

Модулі повинні мати властивості, які представляються адресами й параметрами. Зазвичай ці властивості встановлюються за замовчуванням. Однак у багатьох випадках установлені за замовчуванням значення не відповідають конкретним вимогам. Так, наприклад, заздалегідь установлені види й діапазони вимірів в аналогових модулях чи навряд будуть відповідати бажаним.

Якщо потрібно змінити ці настроювання, дійте в такий спосіб:

1. Двічі клацніть у конфігураційній таблиці на компоненті, що підлягає параметризації, наприклад, на інтерфейсному модулі, або виділіть рядок і виберіть команду меню Edit ⇒ Object Properties (*Редагувати* ⇒ *Властивості об'єкта*).
2. Використовуючи діалогове вікно, що з'явилося, установіть властивості компонента.

Призначення адрес вузлів і входів і виходів

При призначенні адрес слід розрізняти призначення адрес вузлам і призначення адрес входам-виходам.

Адреси вузлів призначаються програмувальним модулям у мережах MPI, PROFIBUS, Industrial Ethernet.

Адреси входів-виходів (I/O) призначаються модулям для того, щоб у програмі користувача зчитувати входи або встановлювати виходи.

Слід урахувати, що адреси MPI для функціональних модулів і комунікаційних процесорів призначаються автоматично. Вони визначаються центральним процесором за наступним правилом:

- перший CP або перший FM після CPU: MPI-адреса CPU + 1;
- другий CP або другий FM після CPU: MPI-адреса CPU + 2 і т.д.

Більш нові CPU S7-300 дозволяють вільне завдання адреси MPI для CP і FM. Адреса встановлюється через закладку General [Загальні (властивості)] модуля.

Адреси входів-виходів в STEP 7 задаються автоматично при розміщенні модулів у конфігураційній таблиці. Кожний модуль має свою початкову адресу (адресу першого каналу). Адреси інших каналів визначаються із цієї початкової адреси, як показано на рисунку 1.5.

Стойка 3	PS	IM (Получатель)	96.0 до 99.7	100.0 до 103.7	104.0 до 107.7	108.0 до 111.7	112.0 до 115.7	116.0 до 119.7	120.0 до 123.7	124.0 до 127.7	
Стойка 2	PS	IM (Получатель)	64.0 до 67.7	68.0 до 70.7	72.0 до 75.7	76.0 до 79.7	80.0 до 83.7	84.0 до 87.7	88.0 до 91.7	92.0 до 95.7	
Стойка 1	PS	IM (Получатель)	32.0 до 35.7	36.0 до 39.7	40.0 до 43.7	44.0 до 47.7	48.0 до 51.7	52.0 до 55.7	56.0 до 59.7	60.0 до 63.7	
Стойка 0	PS	CPU	0.0 до 3.7	4.0 до 7.7	8.0 до 11.7	12.0 до 15.7	16.0 до 19.7	20.0 до 23.7	24.0 до 27.7	28.0 до 31.7	
Слот	1	2	3	4	5	6	7	8	9	10	11

Рисунок 1.5 – Схема адресації входів-виходів у базовій стійці й стійках розширення

Уже використані адреси входів і виходів можна відобразити в такий спосіб:

1. Відкрити станцію, адреси якої потрібно переглянути.
2. Вибрати команду меню View ⇨ Address Overview [*Вид ⇨ Огляд адрес*].
3. Виділити в діалоговому вікні Address Overview модуль, у якому повинні бути відображені призначені входи й виходи, наприклад, CPU.
4. Якщо необхідно, отфільтрувати відображення по видах адрес, наприклад, “тільки адреси входів”.

Адресні області входів і виходів відображаються із вказівкою місця розміщення модулів – номером майстер-системи DP, номером стійки, слота або гнізда. Адреси входів, що мають нульову довжину, наприклад, адреси інтерфейсних модулів, позначаються зірочкою (I*).

На рисунку 1.6 показаний приклад розподілу адрес входів (I) і виходів (Q) для системи S7-400 (CPU 417-4), що складається із центральної стійки (0) і стійки розширення (1). Область адресного простору центрального процесора становить 16383 байта.

У таблиці вікна Address Overview наведені наступні дані (зліва направо):

- тип даних – вхід (I), вихід (Q);
- області адрес кожного модуля (Addr. from і Addr. to);
- найменуванню модуля;
- тип поділу образу процесу (PIP – Process image partition), у якому зазначений організаційний блок OB1 (циклічне виконання);
- адреси в PROFIBUS DP (колонка DP) і PROFINET (колонка PN);
- номер стійки R (Rack) і номер слота S (Slot);
- стартові адреси модулів інтерфейсів DP-master і MPI (колонка IF).

Призначення символічних імен адресам входів і виходів

Уже при конфігуруванні *цифрових або аналогових модулів* їх входам і виходам можна призначити символічні імена, не відкриваючи для цього таблицю символів. Для вбудованих входів-виходів, наприклад, CPU 312 IFM, а також для комунікаційних процесорів CP і функціональних модулів FM символічні імена призначаються через таблицю символів.

При завданні символічного імені необхідно:

1. Виділити цифровий або аналоговий модуль, адресам якого привласнюються символічні імена.
2. Вибрати команду меню Edit (Symbols і у вікні, що відкрилося, внести символічні імена входів або виходів. Якщо клацнути на наявній у діалоговому вікні кнопці Add Symbol, то в якості символу буде внесена адреса операнда.

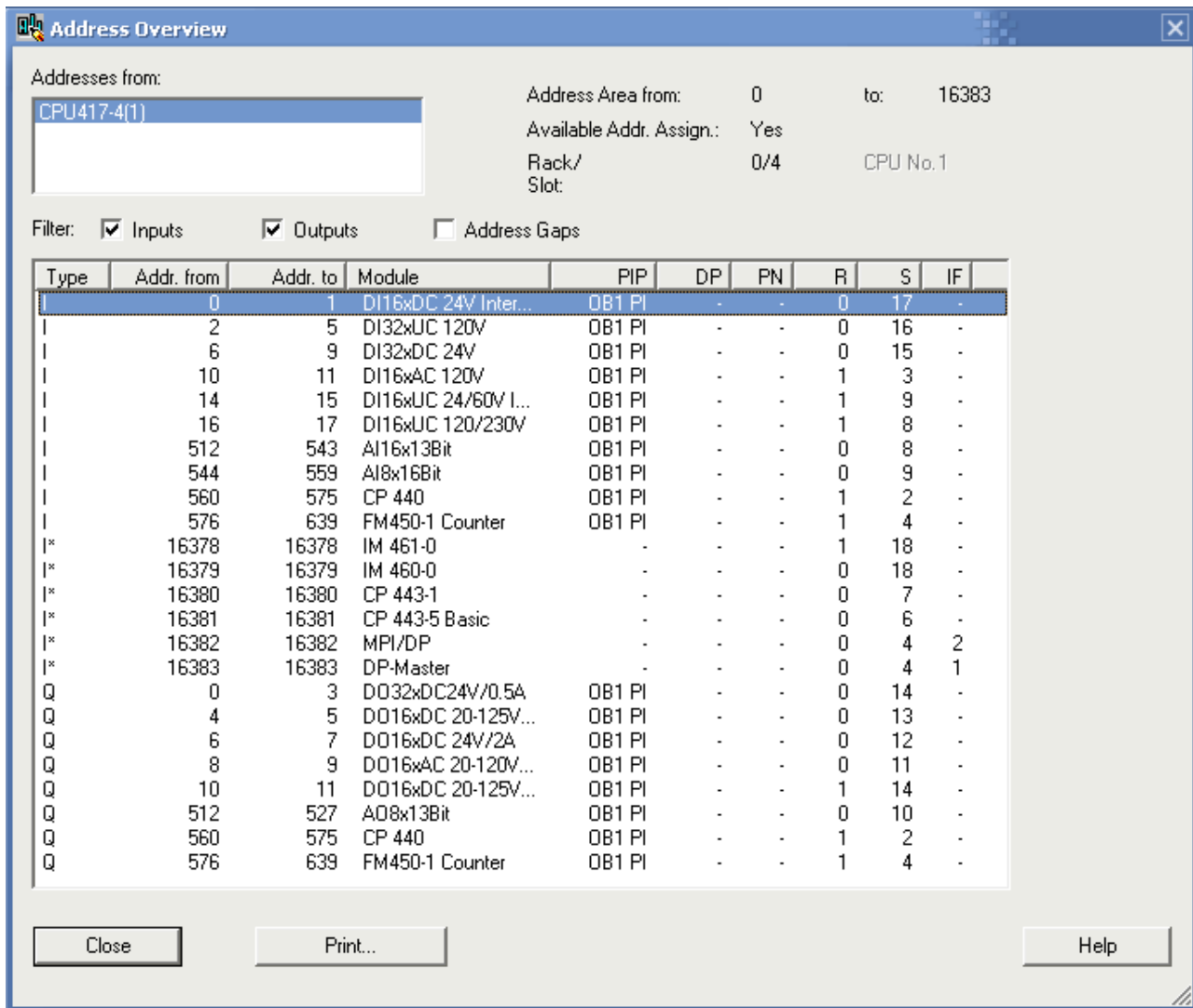


Рисунок 1.6 – Приклад відображення адрес у вікні Address Overview

1.4 Методика виконання індивідуального завдання

Варіанти індивідуальних завдань наведені в Додатку А.

Номер варіанта відповідає порядковому номеру студента в журналі групи.

У завданні втримуються вимоги до центральної станції, що полягає з базової стійки й стійки розширення.

При виконанні роботи необхідно задовольнити наступні вимоги:

1. Обрані блоки живлення повинні забезпечити необхідний струм споживання.
2. Процесорні модулі повинні задовольняти вимогам до комунікацій.
3. При виборі інтерфейсного модуля для з'єднання стійок необхідно визначити доцільність передачі струму в стійки розширення, відстань, кількість стійок розширення, а також необхідність комунікаційної шини.

4. При виборі сигнальних модулів обґрунтувати типи модулів з урахуванням напруг, навантажувальних здатностей і типів з'єднання із зовнішніми пристроями (групування каналів, кількість точок з'єднання, опір навантаження).

5. При виборі комунікаційних процесорів необхідно звернути увагу на тип мережі, підтримувані процесором комунікаційні функції, а також пристрої, з якими цей процесор може взаємодіяти.

У результаті конфігурування центральної станції повинен бути отриманий файл конфігурації з розширенням “.cfg”. Цей файл створюється командою “Station → Export...” у форматі XML. При створенні файлу необхідно вказати директорію для його збереження. Відкрити файл можна в XML Editor.

У звіті необхідно представити:

1. Завдання (варіант).
2. Обґрунтування вибору стійок і модулів.
3. Скріншот вікна Address Overview або його друкований варіант.
4. Файл конфігурації станції.

2 КОНФІГУРУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ ПЕРИФЕРІЇ В МЕРЕЖІ PROFIBUS

Ціль роботи: освоїти методикау й приймання конфігурування й параметрування децентралізованої периферії в мережі PROFIBUS-DP.

2.1 Правила конфігурування децентралізованої периферії

Децентралізована периферія конфігурується в наступній послідовності:

1. На першому етапі створюється мережна структура. Мережна структура є верхнім рівнем організації системи, тому інтерактивний діалог, у якому буде згодом здійснюватися формування нижнього рівня апаратури, проводиться з урахуванням кінцевої мети конфігурації. Саме така послідовність забезпечує правильний вибір модуля центрального процесора й модулів для організації інтерфейсів.
2. На наступному етапі з використанням інтерактивного діалогу вибираються центральний процесорний модуль і засоби комунікації, а також встановлюються властивості й параметри мереж.
3. Робота завершується установкою необхідних сигнальних і технологічних модулів у вузлах мережі.

При конфігуруванні розподіленої периферії необхідно враховувати наступні правила:

1. Усі абоненти мережі повинні мати унікальні адреси. При цьому слід враховувати, що кількість абонентів у мережі MPI повинне бути не більш 32, у мережі PROFIBUS – до 126, а в Industrial Ethernet – до 1024.
2. При установці модуля процесора CPU йому за замовчуванням привласнюється адреса "2". Якщо застосовуються декілька CPU необхідно буде змінювати цю адресу, яка встановлюється за замовчуванням.
3. При призначенні адрес MPI/DP слід враховувати, що адреса "0" резервується для програматора, а адреса "1" – для панелі оператора.

Для відображення мережної конфігурації в NetPro використовуються спеціальні графічні засоби мови конфігурування – символи. Призначення символів показано на рисунку 2.1 (на прикладі мережі MPI).

Символи є елементами керування. При подвійному клацанні лівої кнопки миші на обраному символі відбувається наступне:

- символ станції запускає додаток для конфігурування станції;
- символ підключення до мережі відкриває вікно завдання властивостей інтерфейсу;
- символ модуля відкриває вікно установки параметрів модуля.

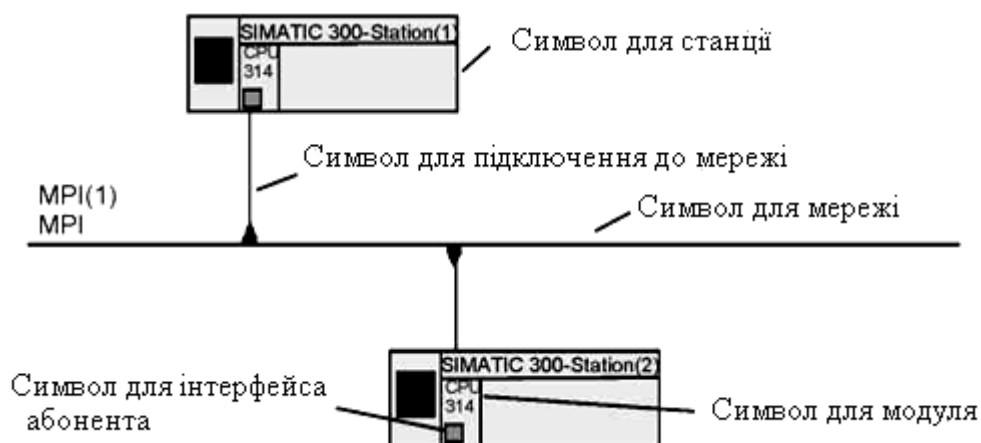


Рисунок 2.1 – Розташування основних символів графічного відображення на прикладі мережі MPI

Зазвичай завдання проектування розподіленої периферії зводиться до забезпечення необхідної організації системи керування й необхідної швидкості передачі даних у мережі, визначенню кількості й типів підмереж, а також формуванню адресного простору.

Як приклад мережної структури на рисунку 2.2 показана система, що складається з інженерної станції ES/OS і активної центральної станції AS, які з'єднані між собою трьома типами інтерфейсів. При цьому центральна станція підтримує дві майстер-системи розподіленої периферії PROFIBUS-DP.

Для забезпечення працездатності системи центральна станція AS постачена модулем центрального процесора CPU 416-2 DP із вбудованим інтерфейсом MPI/DP, а також трьома комунікаційними процесорами CP. Один із цих CP забезпечує зв'язок з інформаційною системою підприємства Industrial Ethernet, а два інших підтримують дві підмережі PROFIBUS з різними профілями налаштувань – DP і Standard.

Профіль DP – це система налаштування параметрів, яку доцільно використовувати в мономастерних системах з еквідістантним циклом шини, а профіль Standard застосовується зазвичай для роботи з мультипроцесорними системами, що вимагають узгодження швидкодії через *різну тривалість* процесів обробки даних.

2.2 Методика створення й параметрування майстер-системи

Майстер-система складається із провідного пристрою (DP-master) і одного або декількох ведених пристроїв (DP-slave).

У якості провідного DP можна використовувати наступні компоненти:

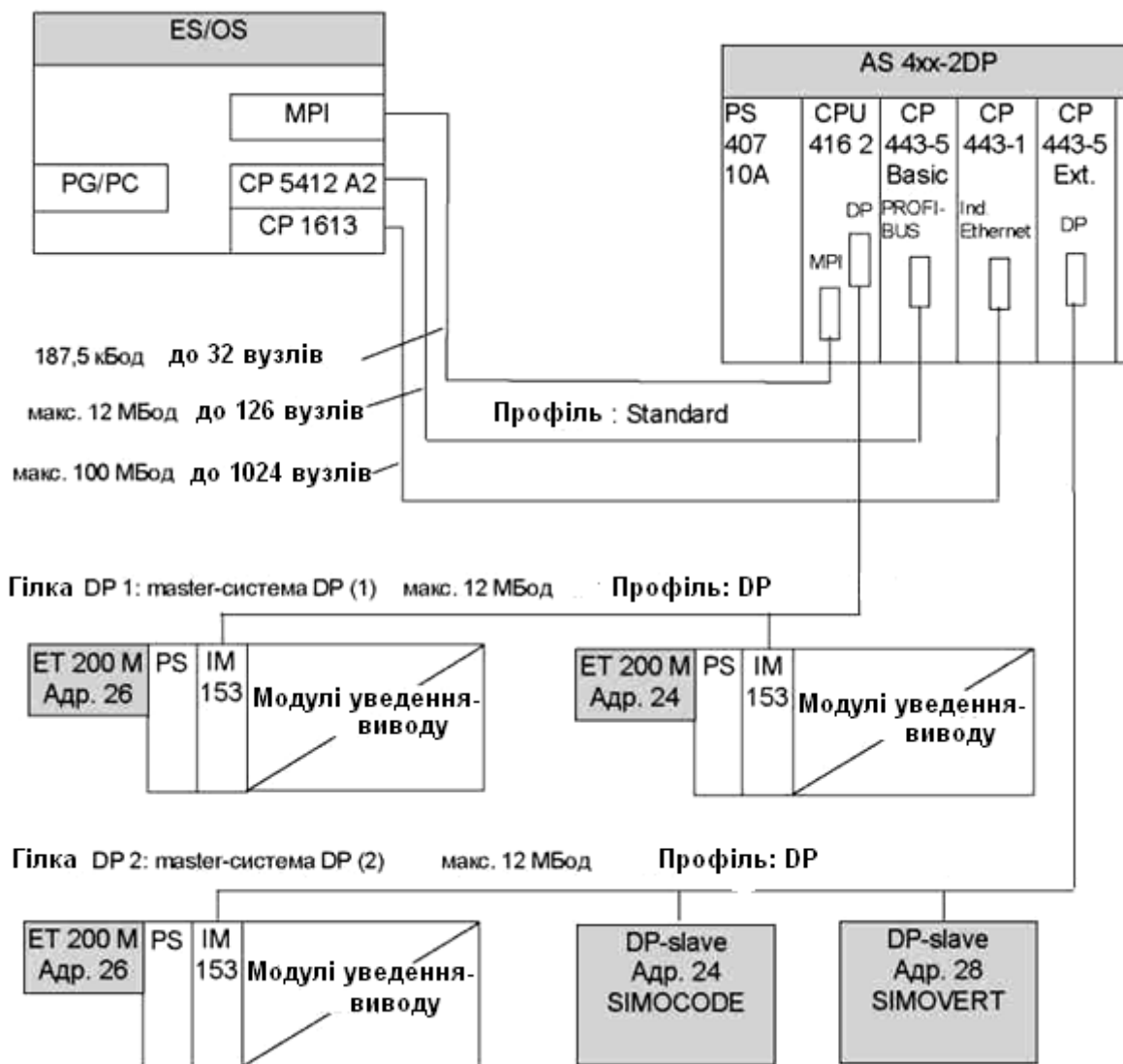


Рисунок 2.2 – Приклад конфігурування системи автоматизації

- CPU із вбудованим інтерфейсом провідного DP;
- інтерфейсний субмодуль, що відповідає обраному CPU, наприклад, IF 964-DP в CPU 488-4;
- комунікаційний процесор CP у поєднанні з CPU, наприклад, CP 342-5 або CP 443-5;
- інтерфейсний модуль із інтерфейсом провідного DP, наприклад, IM 467.

У якості ведених DP використовуються:

- *Компактні ведені DP* – модулі із вбудованими цифровими або аналоговими входами й виходами, наприклад, ET 200B.
- *Модульні ведені DP* – модулі із вбудованими інтерфейсами розширення, наприклад, ET 200M.
- *Інтелектуальні ведені I-Slaves* – це станції, оснащені процесорними модулями й своїми користувацькими програмами, наприклад, S7-300 або ET 200X с BM 147/CPU.

Процес створення майстер-системи рекомендується **вести** в наступній послідовності.

1. Запустіть SIMATIC Manager. У діалоговому вікні установки виду проекту виберіть Finish і в новому вікні (S7_Proj*) уведіть ім'я проекту.

2. Виберіть у меню мережну виставу NetPro. У вікні конфігурування мережі Network за замовчуванням установлена мережа MPI і станція S7-300. Якщо необхідна станція S7-400, то перетягніть її з розділу каталогу у вікно Network, вилучивши перед цим із проекту станцію S7-300.

3. У контекстному меню виберіть команду Object Propertis і в діалоговому вікні Propertis на вкладці General уведіть ім'я станції, наприклад, DP-master. На вкладці Interface установіть типи інтерфейсів, які повинна підтримувати система – MPI, PROFIBUS, Industrial Ethernet, PtP. Закрийте вікно Propertis, підтвердивши зроблені установки кнопкою ОК.

4. У контекстному меню виберіть команду Open Object. Перехід у вікно конфігурування станції вимагає збереження в пам'яті мережної вистави. Підтвердіть свою згоду кнопкою ОК.


5. У порожньому вікні по імені станції (DP-master) конфігурування станції треба почати з установки необхідної стійки. Викличте контекстне меню, виберіть команду Insert Object і в діалоговому інтерактивному режимі виберіть SIMATIC 400 і стійку, наприклад, UR1.

6. Для вибору центрального процесорного модуля також використовуйте інтерактивний діалог, який запускається командою Insert Object. По закінченню процедури вибору STEP-7 виводить вікно Propertis для установки параметрів інтерфейсу PROFIBUS. На вкладці Parameters за замовчуванням установлена адреса “2”, яку можна не міняти.

7. На цій же вкладці для створення майстер-системи натисніть кнопку New і в новому вікні властивостей на вкладці General уведіть ім'я підмережі, а на вкладці Network Setting установіть необхідну швидкість обміну даними по шині, наприклад, 12 Мбіт/с. Тут же можна встановити профіль шини PROFIBUS (DP, Standard, Universal і User Defined).

8. Для завдання постійного часу циклу шини (еквідистантності шини) натисніть кнопку Options і встановіть цикл шини (Constant Bus Cycle Time), наприклад, рівним 10 мс. Підтвердіть установки кнопкою ОК.

9. При установці комунікаційного процесора для організації мережі Industrial Ethernet виберіть тип CP – CP 443-1.

У результаті виконаних дій на відображенні стійки станції створюється рознімання DP (X2) і з'явиться символ майстра-системи . Цей символ є “якорем” для ведених модулів майстер-системи DP.

Відображення створеної станції показано на рисунку 2.3.

Якщо символ майстер-системи у вікні станції не видно, то він, можливо, закритий конфігураційною таблицею. Зменшіть висоту конфігураційної

таблиці, у якій установлений ведучий DP. Якщо символ для майстер-системи DP знову не видно, виберіть команду меню Insert ⇒ DP Master System (Вставити ⇒ Майстер-система DP).

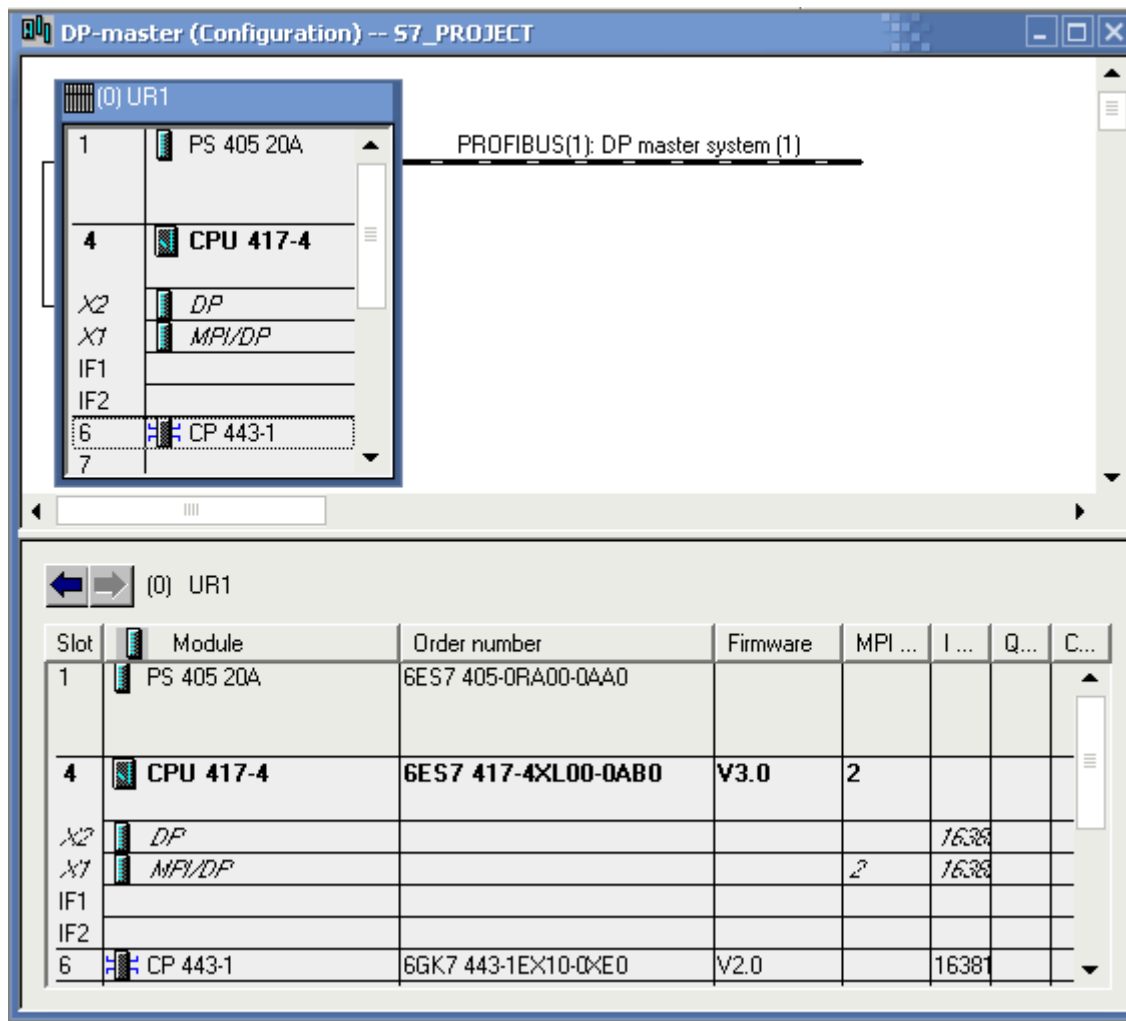


Рисунок 2.3 – Відображення станції майстер-системи у вікні DP-master

Перехід у вікно Netpro дозволяє одержати мережну виставу станції (рис. 2.4).

Після створення провідного DP можна зробити вибір і проектування ведених DP.

Ведені DP, що відповідають установленому провідному модулю, можна перетягувати з вікна каталогу апаратури Hardware Catalog (розділ PROFIBUS-DP) і розміщати на майстер-системі.

При розміщенні ведених модулів DP до майстер-системі DP додається символ, що представляє ведений DP.

Пристаючи до конфігурування ведених модулів, слід урахувати специфічні особливості, пов'язані з конструктивними відмінностями модулів.

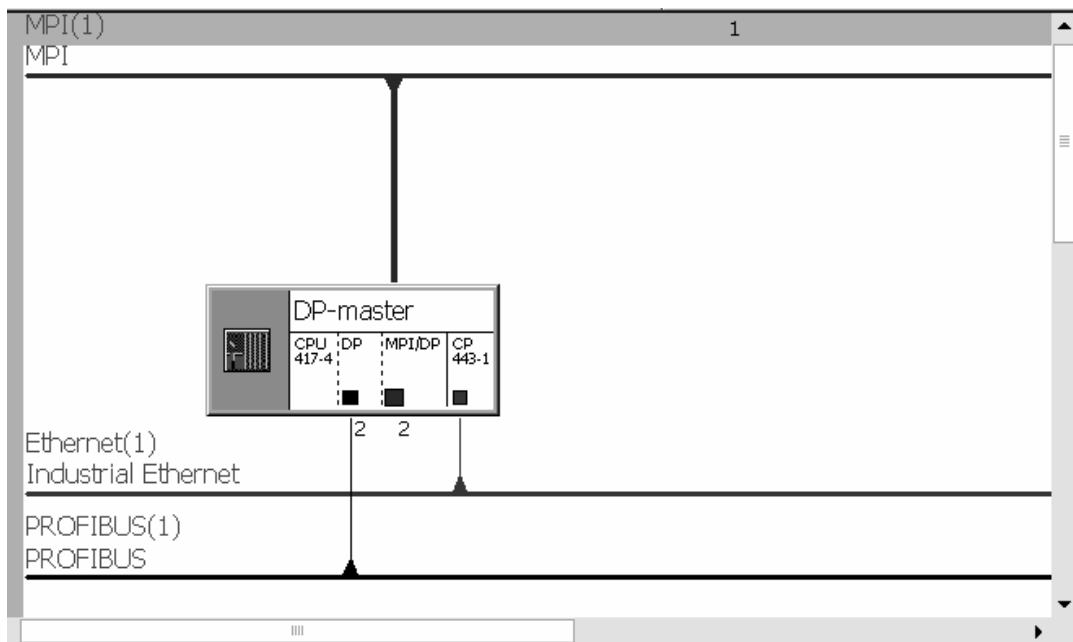


Рисунок 2.3 – Мережна вистава станції у вікні NetPro

2.3 Конфігурування станцій ET 200

Конфігурування станції децентралізованої периферії ET 200M

Модульна станція розподіленого введення-виводу ET 200M комплектується інтерфейсними, сигнальними й функціональними модулями.

Для забезпечення обміну по мережі PROFIBUS DP на станції встановлюється інтерфейсний модуль IM 153. Зв'язок з контролером може здійснюватися також через окремий комунікаційний процесор CP 443-5 Basic.

Станція дозволяє підключити до 8 сигнальних або функціональних модулів. Для живлення модулів станції використовуються або блоки живлення сімейства PS, або блоки живлення сімейства SITOP Power.

При конфігуруванні станції ET 200M слід урахувувати наступні правила до слотів станції:

- Власна периферія (входи/виходи) станції завжди починається зі слота 4.
- Незалежно від того, чи встановлено блок живлення (PS) у реальній структурі чи ні, слот 1 завжди резервується для PS.
- Слот 2 завжди резервується для інтерфейсного модуля DP.
- Слот 3 завжди також резервується для інтерфейсного модуля розширення (IM), незалежно від того, чи є реальний периферійний пристрій розширюваним чи ні.

Правила до слотів необхідно враховувати при конфігуруванні всіх типів ведених DP, як модульних, так і компактних. Призначення слотів важливо для аналізу діагностичних повідомлень, які запускаються слотами.

Станція ET 200M підтримує “гарячу” заміну модулів, тобто заміну без зупинки станції. Для цього сигнальні модулі повинні бути встановлені на активні шинні модулі, які, у свою чергу, монтуються на спеціальну профільну рейку DIN. Активні шинні модулі поєднуються між собою, утворюючи внутрішню шину станції. Якщо “гарячої” заміни не потрібно, сигнальні модулі монтуються на стандартну профільну рейку контролерів S7-300/400.

Приклад.

Нехай майстер-система мережі PROFIBUS DP уже створена й потрібно сконфігурувати ведену станцію ET 200M із одним аналоговим модулем введення, одним аналоговим модулем виводу, а також із цифровим модулем введення й цифровим модулем виводу. Крім того, станція повинна забезпечити рахунок імпульсів з інкрементного датчика, а також з'єднання з AS-шиною.

Конфігурування станції ET 200M здійснюється в наступній послідовності:

1. Виберіть IM 153-2, що забезпечує “гарячу” заміну модулів (папка PROFIBUS DP ⇒ ET 200M) і відбуксируйте цей модуль на DP master system. При цьому STEP-7 виводить діалогове вікно Properties – PROFIBUS node ET 200 IM 153-2 (*Властивості – вузол PROFIBUS ET 200 IM 153-2*). У полі PROFIBUS Address (*Адреса PROFIBUS*) виберіть адресу для slave-пристрою DP, наприклад, 3. Закрийте діалогове вікно кнопкою ОК.

2. Вставка модулів. Розкрийте дерево каталогу (папка PROFIBUS DP / ET 200M / IM153-2) і, буксируючи потрібні модулі з каталогу, вставте їх у слоти ET 200M, починаючи зі слота 4. Функціональний модуль FM 350 для рахунку імпульсів і комунікаційний процесор для зв'язку із шиною AS перебувають у цій же папці.

Результати конфігурування станцій представлено на рисунку 2.4.

Особливості конфігурування станції ET 200S

Станція ET 200S призначена для побудови систем розподіленого введення-виводу на основі PROFIBUS DP або PROFINET.

Станція ET 200S може комплектуватися:

- 1 Звичайними або *інтелектуальними* інтерфейсними модулями для підключення до електричних або оптичних каналів PROFIBUS.
- 2 Модулями введення-виводу дискретних і аналогових сигналів.
- 3 Технологічними модулями для розв'язку завдань позиціонування, швидкісного рахунку, обміну даними через послідовні інтерфейси.
- 4 Силowymi модулями для керування споживачами 3-фазного змінного струму, наприклад, 3-фазними електродвигунами.

Приклад комплектації станції ET 200S показано на рисунку 2.5.

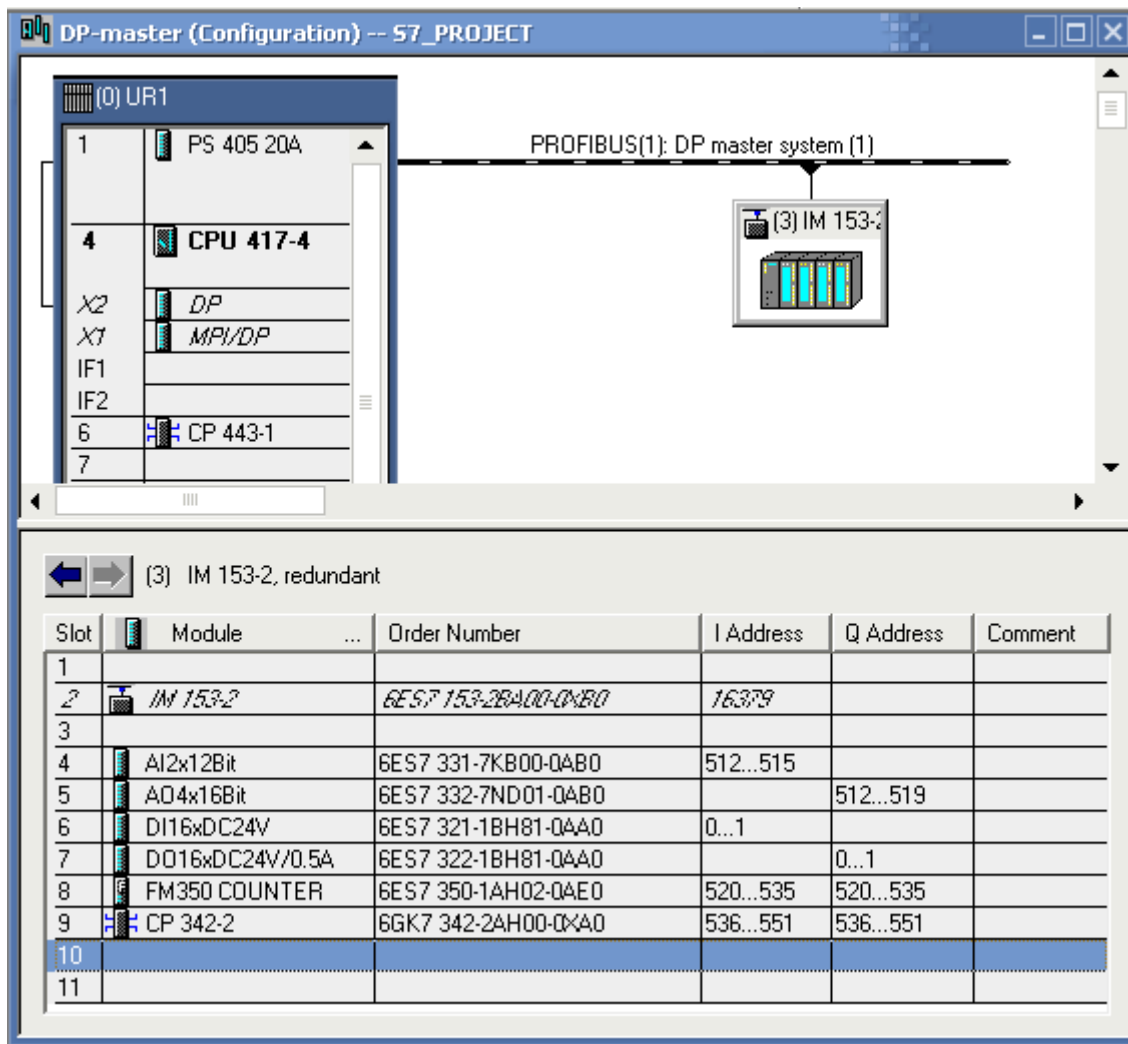


Рисунок 2.4 – Приклад конфігурування станції ET 200M

При конфігуруванні станції ET 200S слід урахувати правила розміщення модулів:

- Першим зліва встановлюється інтерфейсний модуль IM 151.
- справа від інтерфейсного модуля встановлюється модуль контролю живлення електронних модулів PM-E.
- справа від модуля PM розташовуються електронні модулі (ІМ), а також технологічні модулі (FM), які виконують функції рахунку, генерації імпульсів, позиціонування й т.п. Електронні й технологічні модулі встановлюються поверх термінальних модулів ТМ-Е.

Після електронних і технологічних модулів встановлюються засоби мотор-стартера – спочатку модуль контролю живлення силових модулів PM-D, а справа від нього один із стартерів:

- на термінальний модуль ТМ-D, призначений для нереверсивного привода встановлюється силовий модуль (direct starter) DS1-x нереверсивного керування;
- на термінальний модуль ТМ-R, призначений для реверсивного привода, встановлюється силовий модуль RS1-x реверсивного керування.

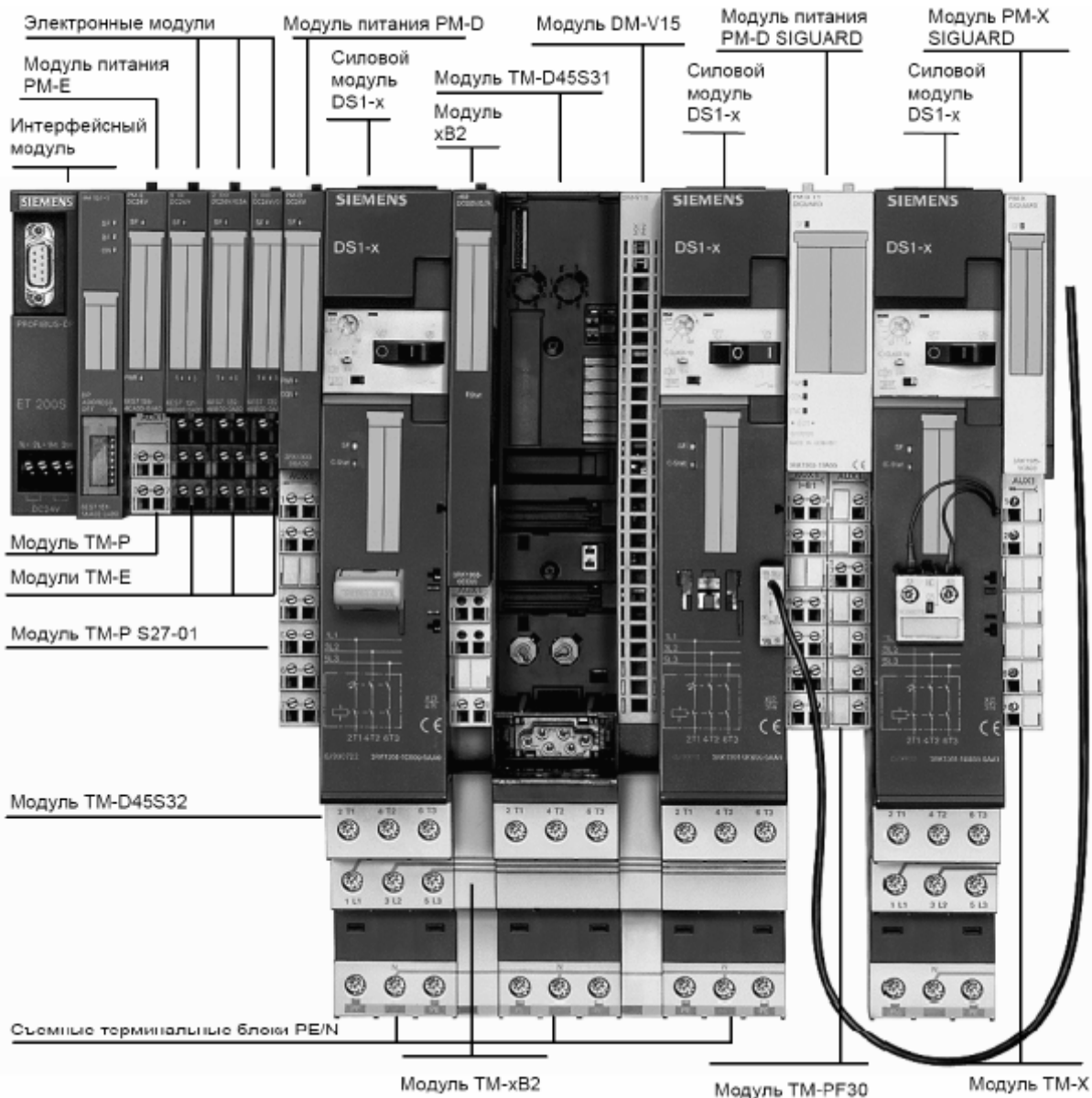


Рисунок 2.5 – Приклад комплектации станции ET 200S

Якщо навантаження вимагає чотирьохпроводного силового ланцюга, силові модулі забезпечуються знімними термінальними блоками PE/N, що дозволяють сформувати нульове проведення N.

Слід урахувати, що кожна *силова група* вимагає окремого модуля контролю живлення PM-D, установлюваного на термінальний модуль TM-P15S27-01.

Станція допускає установку на один інтерфейсний модуль максимум до 63 модулів.

2.4 Конфігурування інтелектуальних ведених DP

Ознакою інтелектуального веденого DP є те, що вхідні й вихідні дані надаються в розпорядження провідному DP не безпосередньо від реального входу-виходу, а від виконуючого попередню обробку CPU, який разом із CP утворює ведений DP.

Інакше кажучи, якщо застосовується звичайний ведений DP, наприклад, компактний ET 200В або модульний ET 200М, то провідний DP звертається до децентралізованих входів-виходів модуля, а якщо застосовується інтелектуальний ведений DP, то провідний звертається не до входів-виходів веденого DP, а до *області операндів* CPU, який виконує попередню обробку. Обмін даними з областю операндів забезпечується програмою користувача CPU, яка виконує обробку даних.

Схема взаємодії провідного й інтелектуального веденого модуля наведено на рисунку 2.6.

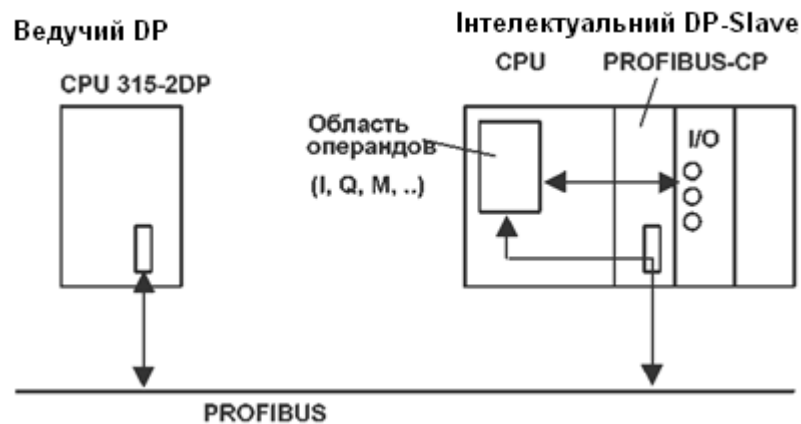


Рисунок 2.6 – Схема взаємодії провідного й веденого інтелектуального модулів

Приклад конфігурування S7-300, як інтелектуального веденого

Нехай потрібно створити наступну конфігурацію:

- Провідна станція (ім'я "DP Master") з CPU 416-2 DP і вбудованим інтерфейсом DP.
- Ведена станція (ім'я "DP Slave") з CPU 315-2 DP у якості інтелектуального веденого DP.

Процес конфігурування містить у собі наступні кроки:

Крок 1. Створюємо проект із іменем S7_Pro_1 із двома станціями – SIMATIC 400 і SIMATIC 300 (рис. 2.7). Клацаємо в дереві проекту по рядку SIMATIC 400, потім по символу Hardware і в порожньому вікні, що відкрилося (SIMATIC 400) командою Insert Object виводимо інтерактивний діалог, у якому спочатку вибираємо SIMATIC 400, а далі стійку UR2.

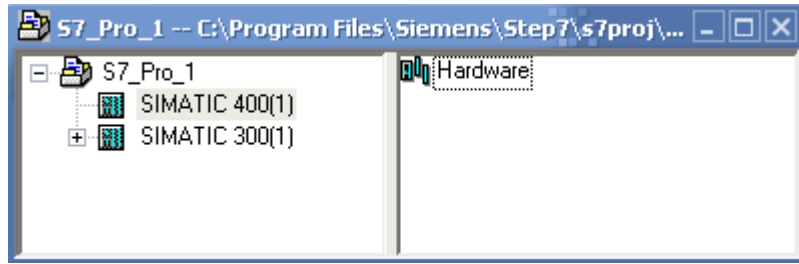


Рисунок 2.7 – Створення проекту

Крок 2. Створюємо провідну станцію SIMATIC 400 і мережу PROFIBUS. Для цього спочатку вибираємо джерело живлення PS 407 10A для слота 1. Для слота 3 по ланцюжку в списку: CPU 400 ⇒ CPU 416-2 ⇒ 6ES7 416-2XK01-0AB0 ⇒ V3.0 вибираємо модуль центрального процесора CPU 416-2. У вікні Properties PROFIBUS interface DP на вкладці Parameters установлюємо адресу – “2”. Далі кнопкою New відкриваємо вікно, у якому задається ім'я мережі – PROFIBUS(1). І, нарешті, на вкладці Network Setting задаємо швидкість передачі по мережі (1,5 Мбіт/с) і профіль (DP).

Крок 3. Створюємо майстер-систему PROFIBUS. Після виконання попереднього кроку у вікні HW Configuration з'являється відображення станції й символ шини: PROFIBUS(1): DP master-system. Клацнувши двічі по рядку слота 4 (вивід X3), у вікні Properties DP на вкладці General у полі Name установлюємо ім'я станції – DP-master, а на вкладці Operating Mode режим – DP-master. У результаті одержимо відображення провідної станції, показане на рисунку 2.8.

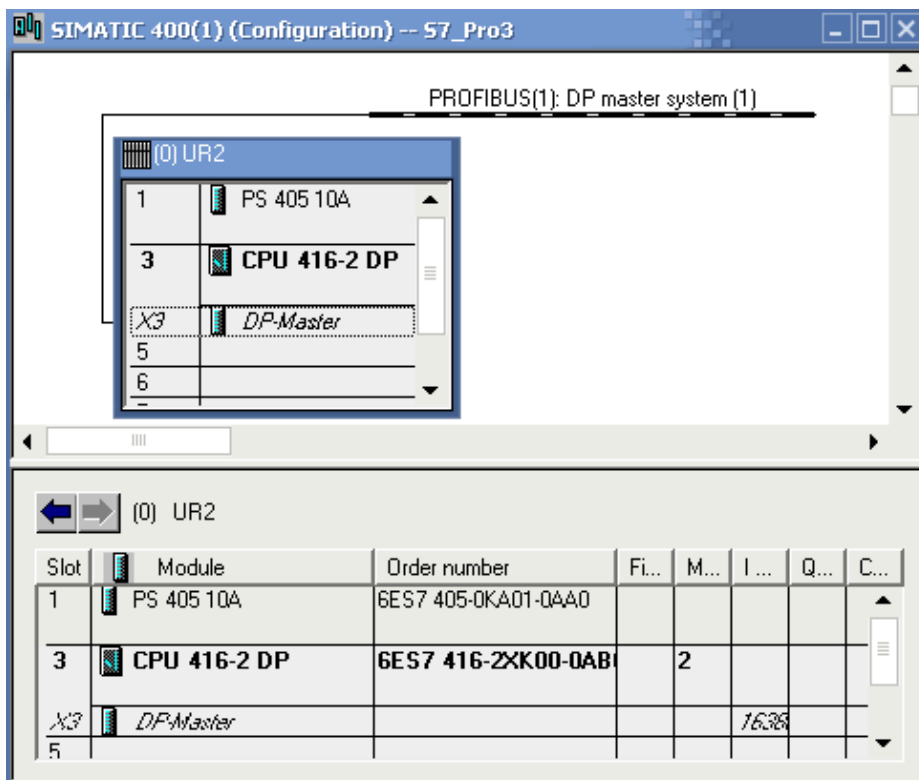


Рисунок 2.8 – Відображення провідної станції PROFIBUS(1)

Крок 4. Створюємо ведену станцію SIMATIC 300 з CPU 315-2 DP. Для цього переходимо в SIMATIC Manager (у вікно проекту, де перебуває станція) і, вибравши станцію у вікні браузера, двічі клацаємо лівою кнопкою миші по її символу Hardware. У результаті відкривається порожнє вікно з іменем станції – SIMATIC 300.

Використовуючи команду Insert Object контекстного меню, встановлюємо стійку (SIMATIC 300 ⇒ RACK-300 ⇒ Rail). Далі робимо тим же способом заповнення стійки модулями – у слот 1 встановлюємо PS 307 2A, а в слот 2 – процесорний модуль CPU 315-2 DP. У вікні Properties вибираємо вже створену шину PROFIBUS(1) і встановлюємо адресу станції в цій шині – 3.

Після цього, клацнувши на рядку слота з інтерфейсом DP, у вікні Properties DP на вкладці General у полі Name встановлюємо ім'я станції – DP-slave, а на вкладці Operating Mode режим – DP-slave.

Результат проектування станцій показано на рисунку 2.9 (відображення у вікні HW Configuration) і на рисунку 2.10 (відображення у вікні Network).

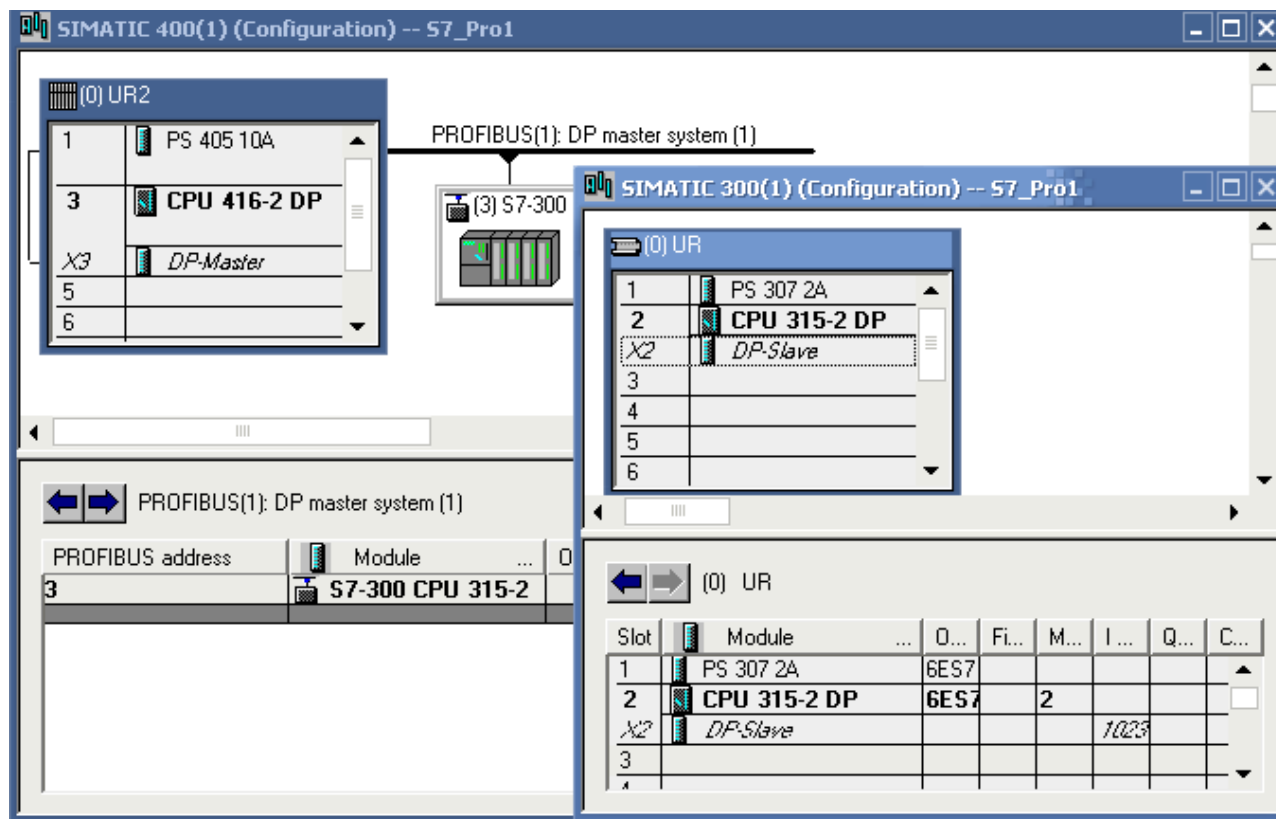


Рисунок 2.9 – Відображення провідної й веденої інтелектуальної станцій у вікні HW Configuration

Крок 5. Встановлюємо з'єднання між станціями. Для цього вертаємося у вікно провідної станції (Windows ⇒ SIMATIC 400). Тут виділяємо символ шини PROFIBUS і командою Insert Object входимо в режим інтерактивного діалогу – вибираємо Configured Station ⇒ CPU 31x. У вікні DP slave properties на вкладці Connect повинна перебувати інформація про ведену станцію, яка

підключається, – ім'я, адреса, тип процесорного модуля й номери слотів. Після натискання на кнопку Connect цей запис повинен зникнути, що свідчить про створення з'єднання.

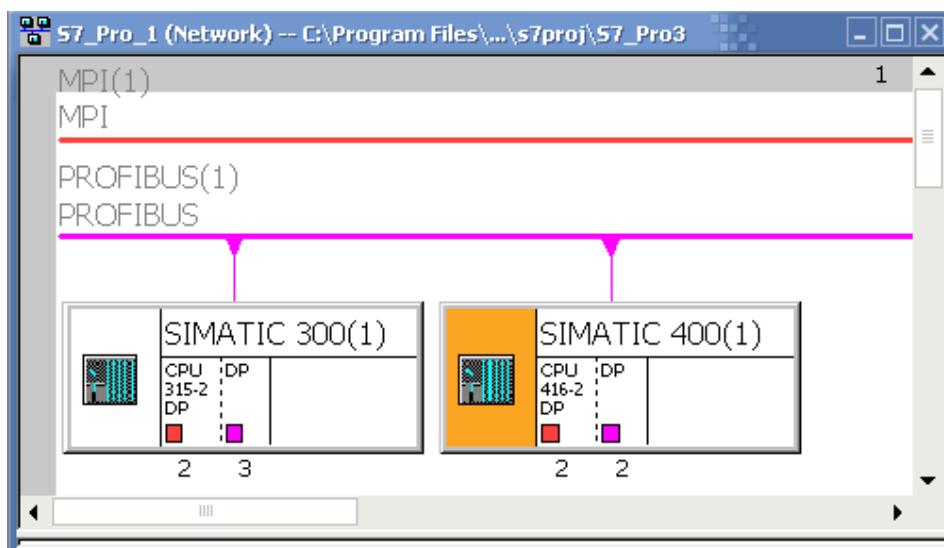


Рисунок 2.10 – Відображення провідної й веденої інтелектуальної станції у вікні Network

Крок 6. Параметруємо з'єднання. У тому ж вікні Properties переходимо на вкладку Configuration і натискаємо на кнопку New. У вікні налаштування параметрів зв'язку (DP slave properties – Configuration – Row 1) є два поля – ліве для провідної станції, праве для веденої. Установивши для DP-master напрямок передачі Output, уведемо в поле адреси адресу пам'яті, з якої дані будуть виводитися, наприклад, 248. Для веденої станції (режим Input) укажемо адресу в області введення, наприклад, 15. Підтверджуємо дані натисканням кнопки OK. У результаті у вікні DP slave properties на вкладці Configuration з'явиться перший рядок даних (Row 1), який визначає властивості з'єднання.

Для параметрування операції читання з веденого пристрою необхідно знову натиснути кнопку New і, змінивши напрямок передачі (в DP-master слід установити Input), потрібно ввести нові значення адрес, наприклад, для ведучого встановити адресу 252, а для веденого – 16.

Підтверджуємо всі установки кнопкою OK. Вид вікна DP slave properties після введення параметрів зв'язку наведено на рисунку 2.11.

Для перевірки правильності конфігурування майстер-системи з інтелектуальним веденим вузлом необхідно виділити одну станцію й вибрати в головному меню Station ⇒ Consistency Check.

Правильність конфігурування оцінюється по консистентності (погодженості) станцій друг щодо друга. Тому така перевірка повинна проводитися для обох станцій. Результати оцінки приводяться у вікні Consistency Check для кожної станції. На рисунку 2.12 вони наведені для станції SIMATIC 400.

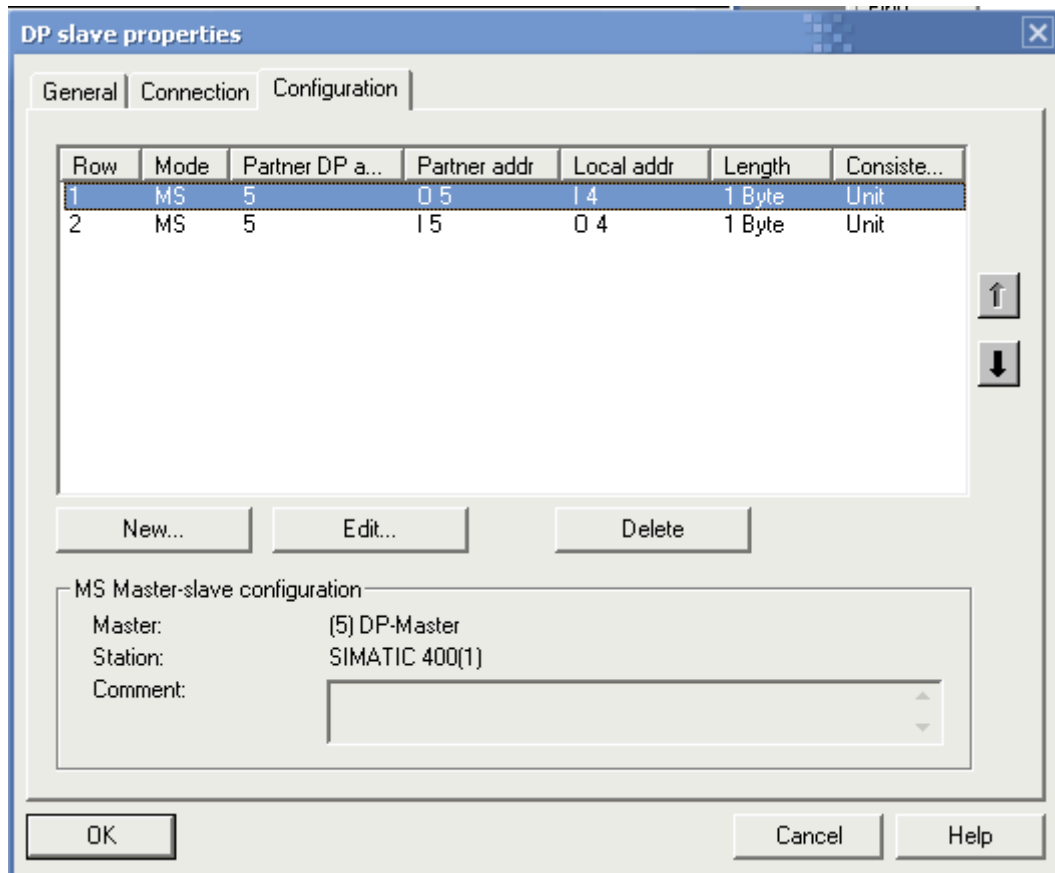


Рисунок 2.11 – Вид вікна *DP slave properties* з параметрами з'єднання

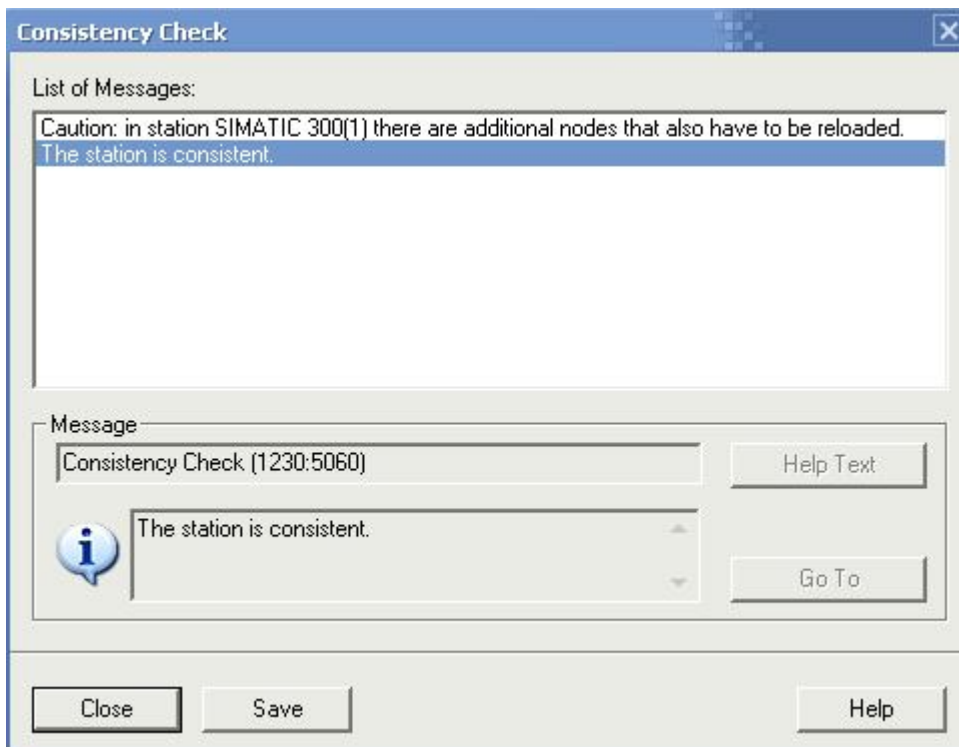


Рисунок 2.12 – Вид вікна з виводом результату перевірки консистентності станцій

Конфігурування ведених станцій з функціями ведучих для підмереж ***Особливості конфігурування DP/ AS-i Link***

При конфігуруванні ведених DP/ AS-i Link (розподілений інтерфейс виконавчих пристроїв і датчиків) слід урахувати, що DP/ AS-i Link конфігурується з веденими AS-i. При розміщенні пристроїв DP/ AS-i Link у нижній частині вікна станції автоматично відображається конфігураційна таблиця, у яку потрібно помістити ведені AS-i з вікна каталогу апаратури Hardware Catalog.

Особливості конфігурування PROFIBUS PA

Щоб сконфігурувати розподілену периферію на польових пристроях PROFIBUS-PA (PROFIBUS для автоматизації процесів), необхідно встановити в мережу PROFIBUS-DP з'єднувач (шлюз) DP/PA.

Конфігурувати з'єднувач DP/PA в утиліті HW Config не потрібно – він невидимий конфігурації станції. Потрібно тільки відбуксирувати DP/PA-link, наприклад, IM 157 з вікна каталогу апаратури на майстер-систему DP. При установці з'єднувача DP/PA виводиться попередження, що швидкість передачі мережі PROFIBUS повинна бути рівна 45,45 Кбод. Для польових пристроїв PA з'єднувач DP/PA зменшить швидкість передачі до 31,25 Кбод.

Відображення DP/PA-link містить у собі поряд із символом для самого пристрою також і символ майстер-системи PA подібно майстер-системі DP. На цей символ і призначаються польові пристрої PA. Однак для цього необхідний додатковий програмний пакет – SIMATIC PDM.

2.5 Методика виконання індивідуального завдання

Варіанти індивідуальних завдань представлені в Додатку Б. При виконанні завдання необхідно зробити наступне.

1. Створити майстер-систему з одним провідним пристроєм.

2. Сконфігурувати провідний пристрій, забезпечивши його, у першу чергу, засобами підтримки розподіленої периферії – центральним процесорним модулем із вбудованим інтерфейсом DP або комунікаційним процесором з автономним виконанням комунікаційних завдань.

3. Сконфігурувати ведені пристрої, зосередивши основну увагу на сумісності інтерфейсних, сигнальних і функціональних модулів, включених у комплектацію цього пристрою.

Результати звіту повинні містити:

- 1) завдання;
- 2) графічна вистава мережі в HW Config і Netpro;
- 3) файли конфігурації кожного вузла мережі.

При захисті роботи необхідно продемонструвати консистентність сконфігурованих вузлів.

3 ПРОГРАМУВАННЯ ЗАВДАННЯ ЛОГІЧНОГО КЕРУВАННЯ

Ціль роботи: освоєння методики й правил розробки програми для розв'язку завдань логічного керування в програмному середовищі STEP 7.

3.1 Інтерфейс редактора LAD/STL/FBD

Більшість завдань логічного керування автоматикою вирішується за допомогою редактора LAD/STL/FBD. Вікно редагування програм на мовах LAD/STL/FBD наведено на рисунку 3.1.

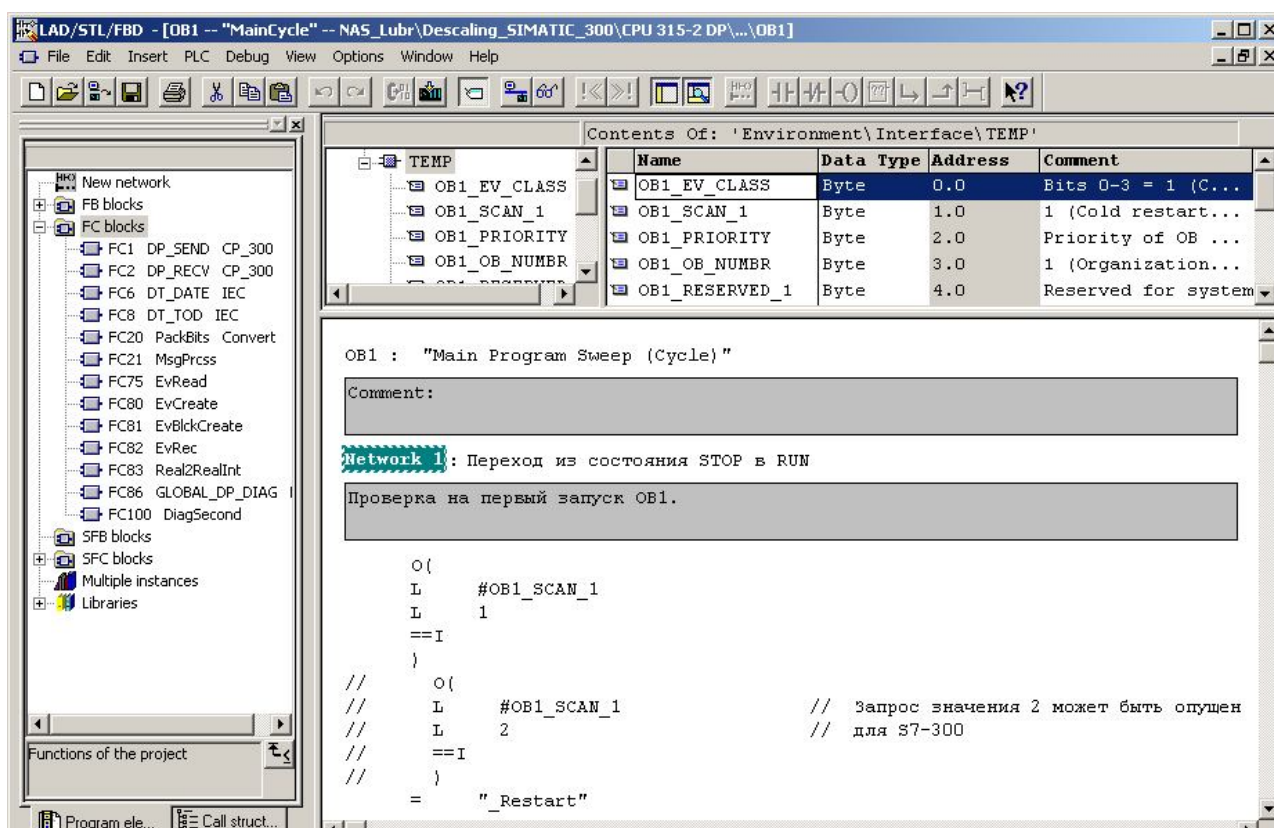


Рисунок 3.1 – Вікно редактора LAD/STL/FBD

У лівій частині вікна розміщується каталог елементів програми. Каталог можна перемістити в праву частину екрана методом Drag&Drop.

Праворуч від каталогу (угорі) розміщена таблиця змінних, а під таблицею – вікно для створення коду програми.

Для установки мови програмування слід відкрити меню View, а для виклику таблиці символів Symbol Table – меню Optins.

Вибір операторів, блоків і функцій проводиться з каталогу елементів.

3.2 Методика створення проекту програми

Завдання на програмування зазвичай містить у собі:

1. Найменування об'єкта керування.
2. Функціональне призначення об'єкта керування.
3. Склад апаратних засобів контролю й керування.
4. Опис процесу керування, вимоги до системи сигналізації й вказівки щодо блокувань.

Для створення проекту необхідно виконати наступне:

1. З урахуванням функціонального призначення й складу встаткування процес керування слід розділити на окремі завдання, які можна реалізувати простими розв'язками. Так, наприклад, у завданнях керування встаткуванням можна виділити окремі виконавчі пристрої, для керування якими потрібна мінімальна кількість датчиків і керуючих сигналів, а в завданнях логічної обробки інформації – етапи перетворення логічних функцій.

2. Для кожного завдання необхідно визначити інтерфейс – скласти список сигналів для входів і виходів, а також побудувати діаграму входів-виходів. Приклади діаграм входів-виходів для двигуна й вентиля наведено на рисунках 3.2 і 3.3.

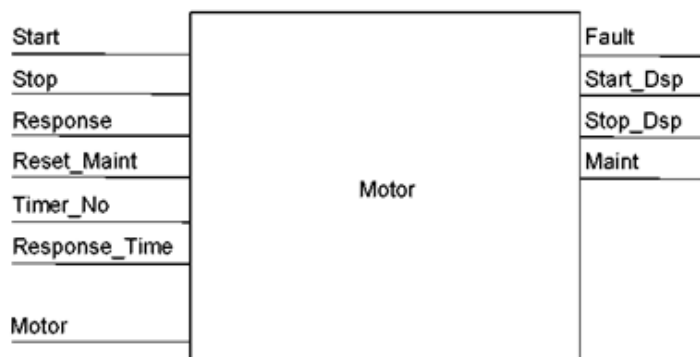


Рисунок 3.2 – Діаграма входів-виходів для двигуна



Рисунок 3.3 – Приклад діаграми входів-виходів для вентиля

3. На третьому етапі необхідно визначити структурну організацію проекту. Кожне завдання вимагає створення функціонального блоку, у якості якого може застосовуватися функція FC або блок FB.

Функції являють собою **параметризуємі блоки без пам'яті**. Функції в класичному виді розширюють набір команд процесора. Проміжні результати, що з'являються під час обробки функції, можуть зберігатися тільки в тимчасових змінних стека локальних даних. Функції застосовуються переважно там, де результат необхідно повертати у блок, який її викликав.

Функціональні блоки (FB) це **кодові блоки з пам'яттю**. Вони можуть викликатися в ОВ, в інших FB і FC. На відміну від функцій (FC) функціональні блоки (FB) мають власну область даних у екземплярному блоці даних DB, де FB може запам'ятовувати стану процесу від виклику до виклику. Функціональні блоки переважно застосовувати в тих випадках, коли потрібно зберігати поточні значення параметрів у цьому ж функціональному блоці для використання в наступних циклах.

На рисунку 3.4 показаний приклад ієрархії блоків, викликуваних у структурованій програмі.

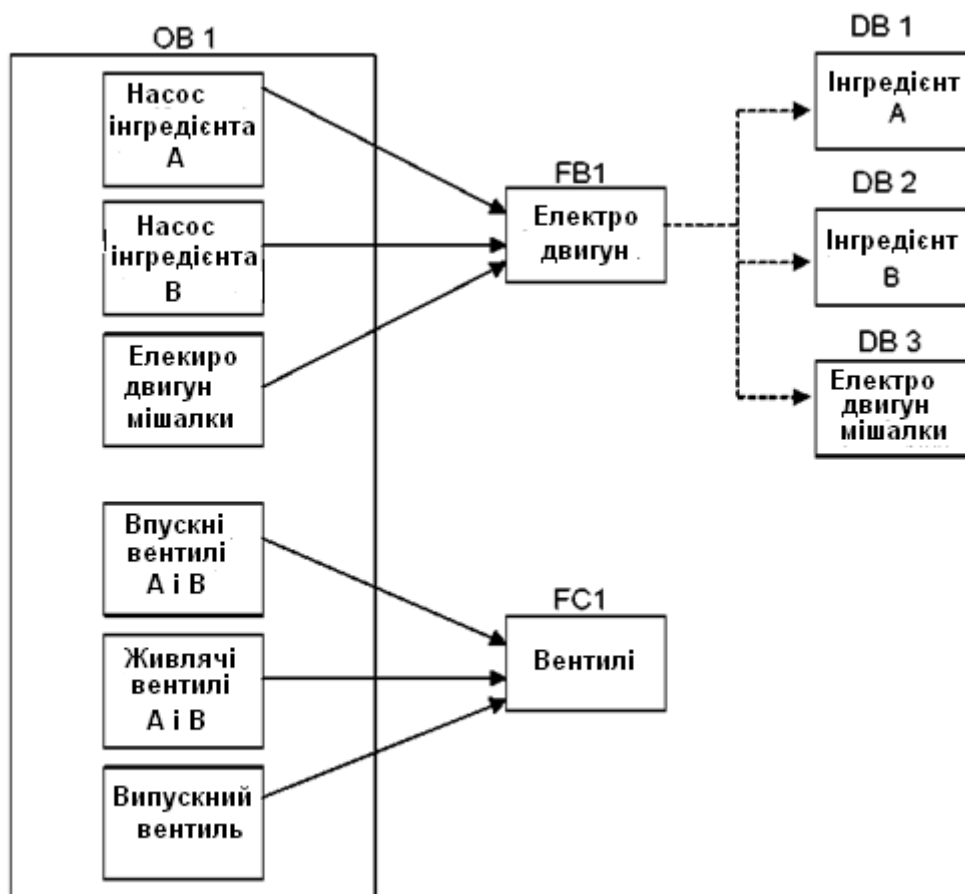


Рисунок 3.4 – Приклад структури програми й схеми викликів блоків

Приклад демонструє структуру програми для керування процесом змішування двох інгредієнтів (А і В). Програма втримується в організаційному

блоці OB1, який утворює інтерфейс із операційною системою CPU. У блоці OB1 викликаються блоки FB1 для керування трьома двигунами насосів і FC1 для керування вентилями. У зв'язку з тим, що статичні дані для керування електродвигунами живильних насосів і мішалки різні, для їхнього збереження застосовується не один, а три екземплярних DB, пов'язаних з FB1.

4. На наступному етапі слід призначити глобальні змінні, які будуть занесені в таблицю символів проекту. У список глобальних змінних слід внести імена всіх блоків і функцій, лічильників і таймерів, а також сигналів помилок, сигналів управління від пульта керування й аварійних сигналів датчиків, які можуть бути використані в будь-якому програмному блоці.

5. Використовуючи відомості про структурну організацію програми й глобальних змінних, слід перейти до призначення локальних змінних, тобто тих змінних, які будуть використовуватися у функціональних блоках. Тут необхідно проаналізувати необхідні блокування й забезпечити умови безпеки, для задоволення яких можливо буде потрібно введення додаткових сигналів і датчиків.

Перераховані вище етапи недостатні для переходу до програмування, тому що перед програмуванням блоків глобальні й локальні змінні повинні бути оголошені. При цьому повинні бути зазначені їхні адреси. Однак призначення адрес можливо тільки тоді, коли відома конфігурація контролера. Тому наступним етапом створення проекту є конфігурування системи автоматизації. У результаті програміст одержує важливі документи – схеми підключення модулів вводу-виводу.

Програма повинна розміщатися в робочій пам'яті центрального процесорного модуля (CPU) контролера. Інтерфейс цієї програми з операційною системою CPU забезпечують організаційні блоки OB. Для циклічного виконання програми вона повинна бути поміщена в організаційний блок OB1. Слід урахувати, що спочатку створюються блоки, які не викликають ніяких інших блоків. Далі блоки створюються за правилом: викликувані в них функції й блоки повинні бути вже створені. Відповідно до цього правила організаційний блок OB1 повинен створюватися останнім.

Розробка програми починається із вставки в проект компонента "S7 Program". При цьому створюється порожня таблиця символів для глобальних змінних, яку можна відкрити в меню Options. У таблицю вносяться імена всіх блоків і функцій, а також глобальні змінні.

Після заповнення таблиці символів можна перейти до створення блоків і функцій. Для цього в меню Insert вибирається команда "S7 Block", яка відкриває список із шести можливих типів блоків, функцій і таблиць.

Після вставки функціонального блоку в програму користувача "S7 Program" необхідно вибрати мову програмування, а у вікні оголошення змінних редактори оголосити всі параметри введення-виводу, а також статичні

й тимчасові змінні. Для всіх функціональних блоків повинні бути створені блоки даних.

Робота створеної програми повинна бути перевірена в програмному додатку S7-PLCSIM. Після перевірки програми проводиться її компіляція й збереження.

Таким чином, створенню програми передують значна підготовча робота.

3.3 Приклад програмування системи керування

Найменування об'єкта

Стрічковий конвеєр.

Функціональне призначення

Стрічковий конвеєр призначений для передачі виробу від одного робочого місця до іншого. Для обслуговування декількох робочих місць створюється відповідна кількість конвеєрів.

Склад апаратних засобів контролю й керування

Стрічковий конвеєр приводиться в рух електродвигуном, що одержує живлення від перетворювача напруги.

Керування здійснюється за допомогою пульта, постаченого кнопками запуску системи керування й переключення системи в початковий стан, кнопками включення й останову двигуна транспортера, перемикачем режиму роботи (автоматичний і ручний), а також декадним перемикачем завдання кількості виробів у партії.

У якості засобів контролю застосовані датчики типу "світловий бар'єр" для контролю наявності виробу на стрічці транспортера в початковій і кінцевій позиціях, датчик швидкості обертання двигуна, а також запобіжний датчик для контролю живлення двигуна.

Опис процесу керування, вимоги до системи сигналізації й вказівки щодо блокувань

Основні функції процесу керування представлені нижче:

- Установка контролера у початковий стан здійснюється сигналом "Basic_st".
- Якщо в точці завантаження на стрічці конвеєра деталей немає, то контролер сповіщає про це сигналом "Readyload" ("готовий до завантаження").
- Включення руху стрічки конвеєра для транспортування деталей здійснюється сигналом "Start" ("запуск"), а в ручному режимі – сигналом "Man_on". Керуючий сигнал на двигун має ім'я "Bel_mot_on".
- По сигналу "Continue" ("продовжити") стрічка конвеєра з деталями продовжує рух, поки датчик "end-of-belt" ("кінець конвеєра") не виявить присутність деталей.

• Якщо датчик, наприклад, фотоелемент, "End-of-belt" ("кінець конвеєра") на кінцевій ділянці конвеєра виявляє присутність деталей, то контролер установлює вихідний сигнал "Ready_rem" ("готовий до вивантаження") і зупиняє мотор конвеєра сигналом.

• Запобіжному вимикачу двигуна привласнене найменування "Mfault".

• Статичні змінні містять у собі: сигнали завантаження нових деталей на транспортер "Load" і вивантаження деталей "Remove", а також меркери позитивних і негативних фронтів цих сигналів – "EM_Rem_P", "EM_Rem_N", "EM_Loa_P", "EM_Loa_N", відповідно.

Нижче наведена таблиця символів "Symbols" (рис. 3.5) і програма керування транспортером, що представляє собою функцію FC 11 мовою STL.

Status	Symbol	Address	Data type	Comment
	Belt_control	FC 11	FC 11	Система управления конвейером
	Basic_st	I 0.0	BOOL	Установка контроллеров в исходное состояние
	/Stop	I 0.2	BOOL	Остановка двигателя привода конвейера ("zero-active" ["активный ноль"])
	Start	I 0.3	BOOL	Запуск конвейера
	Continue	I 0.4	BOOL	Подтверждение того, что деталь была снята с конвейера
	Light_barrier1	I 1.0	BOOL	Фотоэлемент, датчик "End of belt" ("Конец конвейера") для конвейера №1
	Readyload	Q 4.0	BOOL	Загрузка новых деталей на транспортер ("Готов к загрузке")
	Ready_rem	Q 4.1	BOOL	Выгрузка деталей с транспортера ("Готов к выгрузке")
	Man_on	I 0.1	BOOL	Включение двигателя привода конвейера
	/Mfault1	I 2.0	BOOL	Предохранительный выключатель двигателя
	Belt_mot1_on	Q 5.0	BOOL	Управляющий сигнал на включение двигателя конвейера №1
	Load	M 2.0	BOOL	Команда загрузки новых деталей на транспортер
	Remove	M 2.1	BOOL	Команда выгрузки деталей с транспортера
	EM_Rem_N	M 2.2	BOOL	Меркер фронта (отрицательного) "remove" ("выгрузка детали")
	EM_Rem_P	M 2.3	BOOL	Меркер фронта (положительного) "remove" ("выгрузка детали")
	EM_Loa_N	M 2.4	BOOL	Меркер фронта (отрицательного) "load" ("загрузка детали")
	EM_Loa_P	M 2.5	BOOL	Меркер фронта (положительного) "load" ("загрузка детали")

Рисунок 3.5 – Таблица символів для S7-програми

Програма FC 11 мовою STL

FUNCTION Belt_control : VOID

TITLE := Control of a conveyor belt

VERSION : 01.00

BEGIN

NETWORK 1

TITLE = Load parts

//У цьому сегменті виконується завантаження й запуск транспортера

A Start; //Запуск конвеєра

S Load;

O Light_barrier1; //Деталі досягають кінця стрічки

O Basic_st;

ON"/Mfault1"; //Запобіжний вимикач мотора

R Load;

NETWORK 2

TITLE = Parts ready for removal //Коли деталі досягли кінця

```

                                конвеєра, вони можуть бути зняті
A      Load;                    // При досягненні кінця конвеєра
FN     EM_Loa_N;                //"Load" скидається
S      Ready_rem;              //Деталі можуть бути зняті
A      Remove;
FP     EM_Rem_P;                // Деталі зняті
O      Basic_st;
ON     "/Mfaultl";
R      Ready_rem;

```

NETWORK 3

TITLE = Remove parts

//Команда "Remove" ініціює зняття деталей із транспортера

```

A      Continue;                //Вмикач реверсу конвеєра
S      Remove;
ON     Light_barrierl;          //Деталі зняті зі стрічки
O      Basic_st;
ON     "/Mfaultl";              //Запобіжний вимикач мотора
R      Remove ;

```

NETWORK 4

TITLE = Belt ready for loading

//Коли деталі зняті з конвеєра, можна поставити на стрічку нові

```

A      Remove;
FN     EM_Rem_N;                //Деталі зняті
O      Basic_st;
S      Readyload;              //Стрічка конвеєра порожня
A      Load;
FP     EM_Loa_P;                //Транспортер запущений
ON     "/Mfaultl";
R      Readyload;

```

NETWORK 5

TITLE = Control belt motor

//Двигун привода включається й
//вимикається в даному сегменті

```

A      (
O      Load;                    //Завантаження деталей на транспортер
O      Remove;                  //Видалення деталей із транспортера
O      Map_op;                  //Запуск за допомогою "Map_op" (без
                                //реманентності)
)
A      "/Stop";                 //Зупинка двигуна транспортера
ON     "/Mfaultl";              //Запобіжний вимикач мотора
=      Belt_motorl;

```

NETWORK 6

TITLE = Block end

```

BE
END_FUNCTION

```

Ця ж програма на мові LAD наведена на рисунках 3.6 – 3.10.

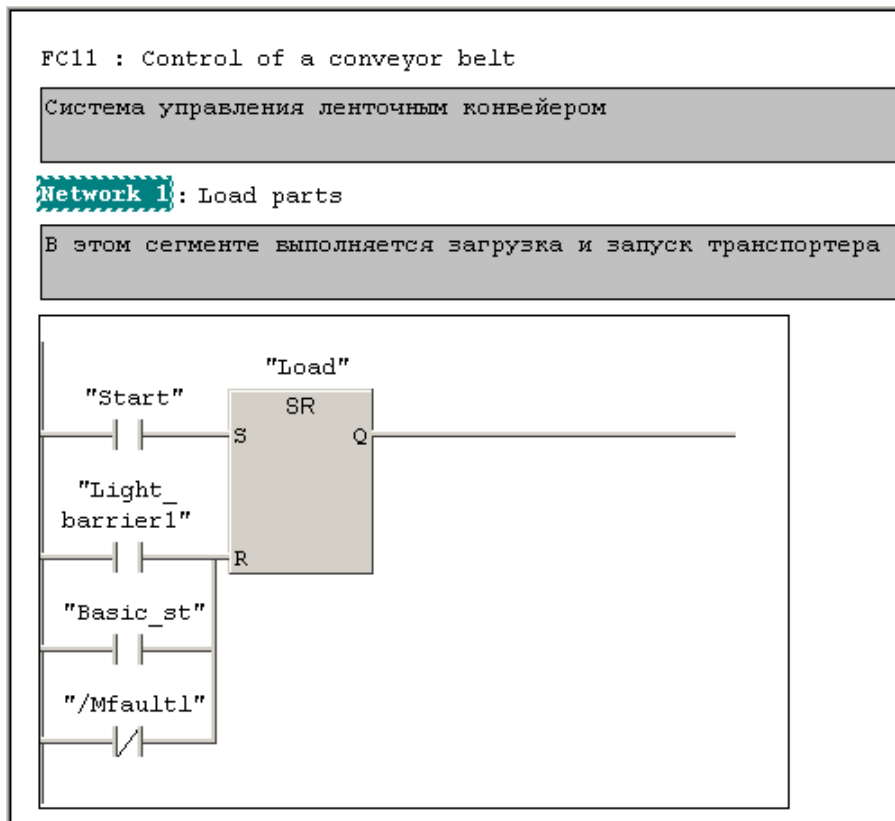


Рисунок 3.6 – Програмний ланцюг завантаження транспортера

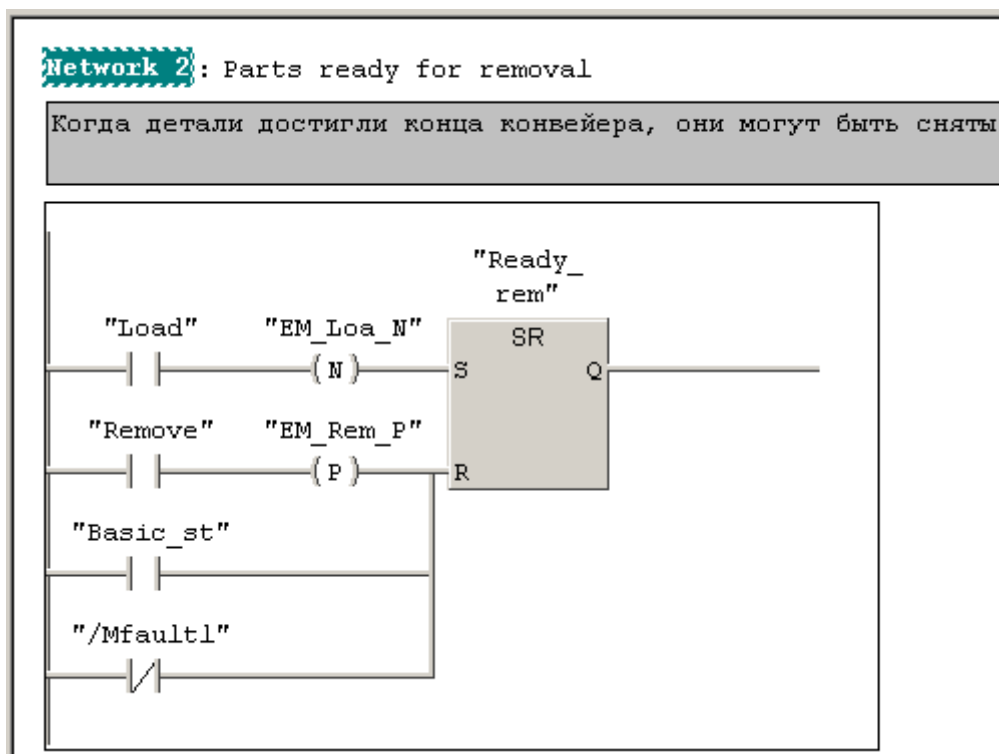


Рисунок 3.7 – Програмний ланцюг, що контролює місце вивантаження транспортера

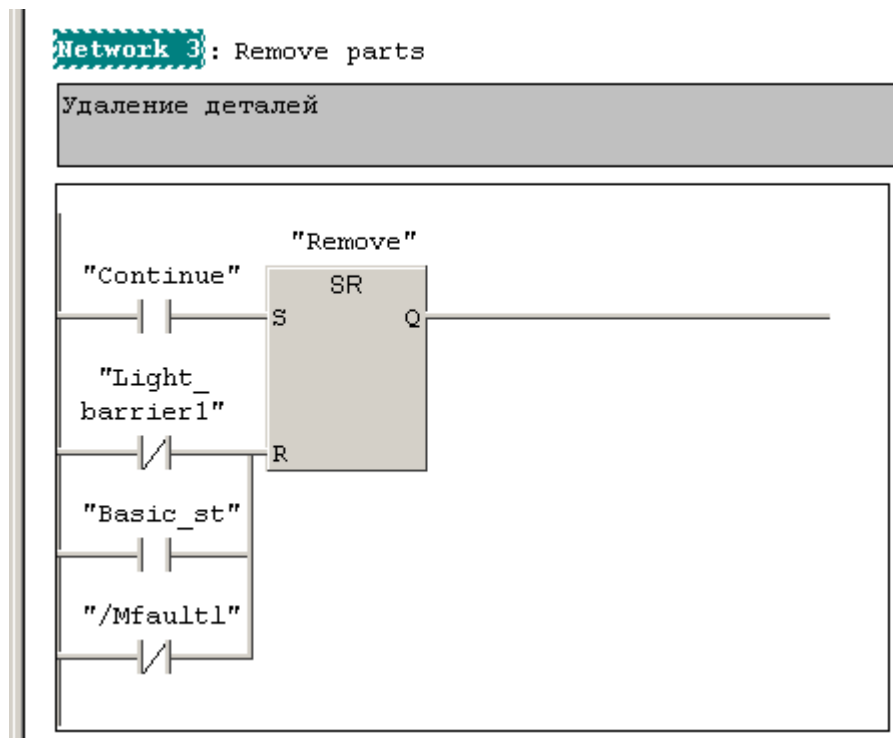


Рисунок 3.8 – Програмний ланцюг вивантаження транспортера

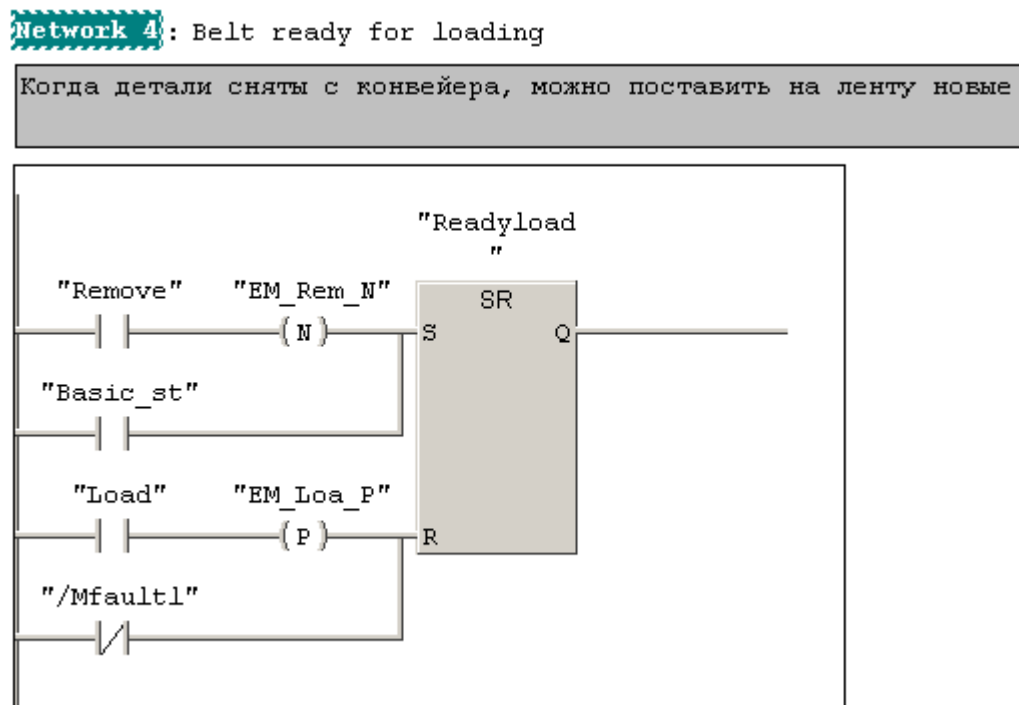


Рисунок 3.9 – Програмний ланцюг, що контролює завантаження транспортера

Програма керування транспортером може бути також реалізована функціональним блоком FB.

Network 5: Control belt motor

В данном сегменте производится управление мотором транспортера

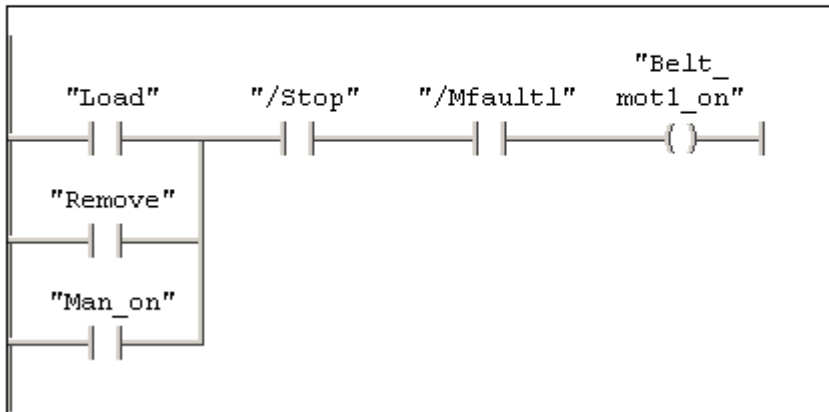


Рисунок 3.10 – Програмний ланцюг керування мотором транспортера

Схема функціонального блоку "Conveyor_belt" системи керування стрічковим конвеєром показана на рисунку 3.11. На схемі показані входи, виходи й статичні змінні, що включають у себе меркери.

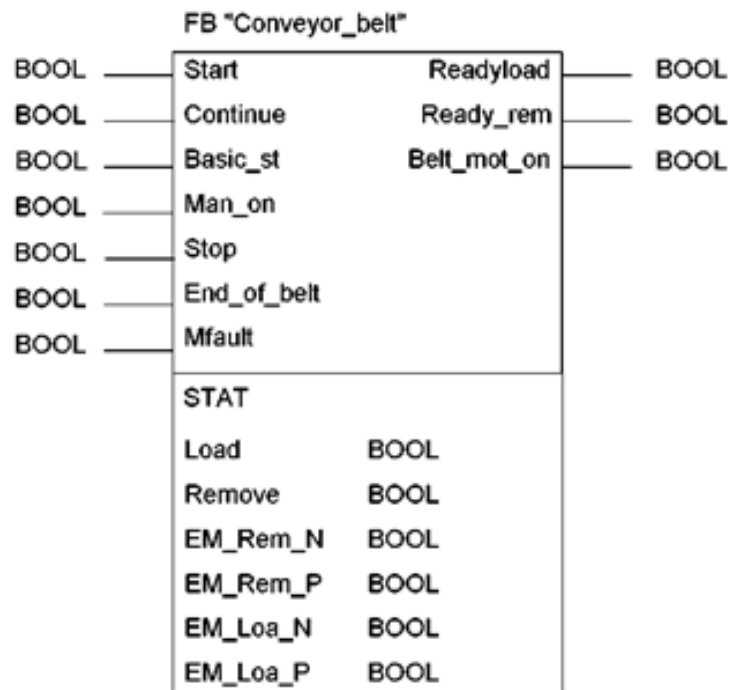


Рисунок 3.11 – Функціональний блок "Conveyor_belt"

Функціональний блок FB "Conveyor_belt" можна використовувати для керування двома і більше стрічковими конвеєрами. Для керування двома конвеєрами блок повинен викликатися двічі: перший раз – для обробки входів і виходів конвеєра 1, другий раз – для обробки входів і виходів конвеєра 2.

Для кожного виклику функціонального блоку потрібен окремий екземплярний блок даних, у якому зберігаються дані відповідного конвеєра. Блок даних для конвеєра 1 можна назвати, наприклад, "Belt_data1", а блок даних для конвеєра 2 – "Belt_data2" і т.д.

Нижче наведений приклад програмування функціонального блоку "Feed" ("подаючий механізм") для керування чотирма стрічковими конвеєрами.

У функціональному блоці "Feed" блок ФВ "Conveyor_Belt" ("Стрічковий конвеєр") повинен викликатися чотири рази. Для кожного виклику ФВ "Conveyor_Belt" призначається свій екземплярний блок, а всі викликувані функціональні блоки зберігають свої дані в екземплярному блоці даних функціонального блоку "Feed".

На рисунку 3.12 показано, як окремі блоки керування конвеєрами з'єднуються в єдину систему керування.

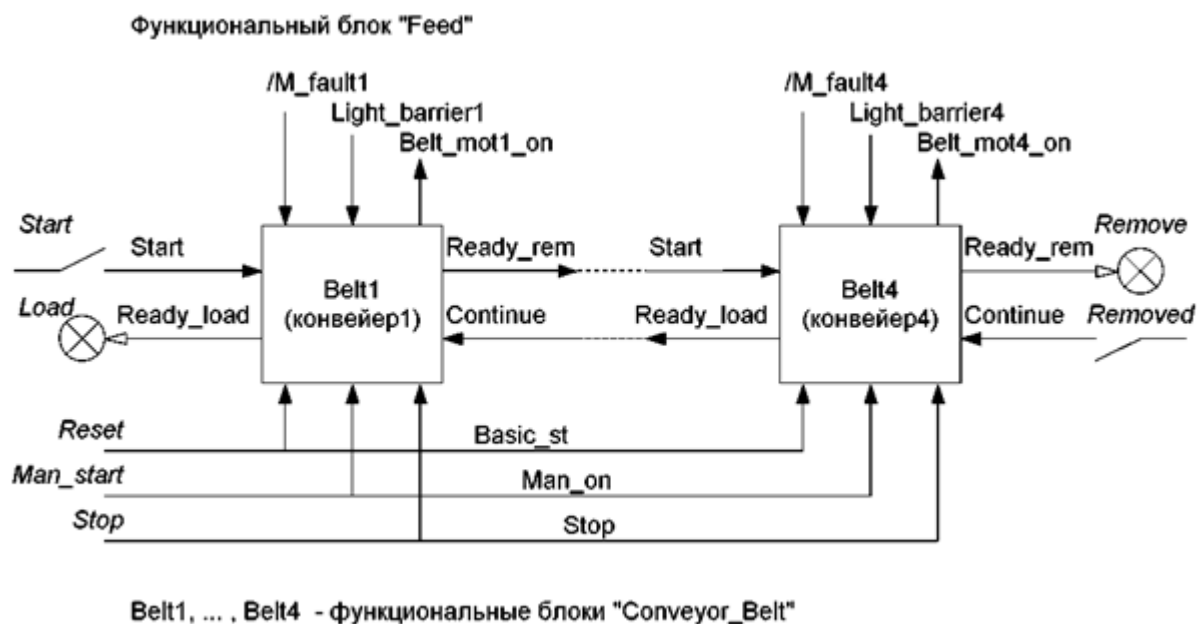


Рисунок 3.12 – Принцип з'єднання функціональних блоків "Conveyor_Belt"

З рисунка 3.12 видно, що сигнал Start подається на вхід Start блоку управління конвеєра Belt 1, вихід Ready_rem подається на вхід Start блоку управління конвеєра Belt 2 і, нарешті, Ready_rem від блоку керування Belt 4 подається на вихід Remove системи керування "Feed". Такий же ланцюжок сигналів проходить у зворотному напрямку: Removed ⇒ Continue ⇒ Ready_load ⇒ ... ⇒ Load.

Сигнали Belt_motx_on, Light_barrierx і /M_faultx (відмова мотора) – це окремі сигнали в кожному із блоків керування конвеєрами. Входи Reset, Man_start і Stop дозволяють управляти всіма блоками за допомогою сигналів Basic_st, Man_on і Stop, відповідно.

Дані окремих блоків керування конвеєрами в розділі статичних локальних даних слід оголосити так, як оголошуються дані користувачького типу UDT, тобто із вказівкою імені й типу даних. Змінна *Belt1* повинна містити структуру даних функціонального блоку "Conveyor_Belt", так само як і змінні *Belt2*, ..., *Belt4*.

Програма функціонального блоку починається з ініціалізації загальних сигналів для всіх блоків керування конвеєрами. Тут ураховується той факт, що параметри блоків даних для функціональних блоків, викликуваних як локальні екземпляри, є статичними локальними даними в поточному блоці. Параметр блоку *Man_start* поточного функціонального блоку управляє вхідним параметром *Man_on* усіх чотирьох блоків керування конвеєрами за допомогою простої операції присвоєння. У такий же спосіб проводиться ініціалізація вхідних параметрів *Basic_st* і *Stop* за допомогою параметрів блоку *Reset* і *Stop* відповідно.

Послідовні виклики функціональних блоків керування конвеєрами містять параметри блоку тільки для окремих сигналів відповідного конвеєра, пов'язані з параметрами функціонального блоку "Feed". Ці окремі сигнали являють собою сигнали від фотодатчиків (*Light_barrierx*), сигнали для моторів і від моторів приводів конвеєрів (сигнали для аварійної зупинки двигунів *M_faultx* і сигнали про стан моторів приводів *Belt_motx_on*).

Програмування зв'язку між окремими блоками керування конвеєрами проводиться з використанням операцій присвоєння.

Програма функціонального блоку "Feed"

```
FUNCTION_BLOCK "Feed"
TITLE = Control of several conveyor belts      //Керування декількома
                                              //стрічковими конвеєрами
//Приклад оголошення локальних екземплярів
NAME : Feed
VERSION : 01.00

//Розділ оголошення змінних

VAR_INPUT
Start : BOOL := FALSE;                       //Запуск стрічок конвеєрів
Removed: BOOL := FALSE;                     //Деталі вилучені зі стрічки
Man_start : BOOL := FALSE;                  //Ручний запуск конвеєрів
Stop : BOOL := FALSE;                       //Зупинка стрічок конвеєрів
Reset: BOOL := FALSE;                      //Установка у початковий стан (basic_st)
Tim : TIMER;                                //Функція таймера
Dura1: S5TIME := S5T#5s;                    //Контрольний час для деталей
Dura2 : S5TIME := S5T#10s;                 //Контрольний час для перерви
END_VAR

VAR_OUTPUT
Load:      BOOL      := FALSE;              //Завантаження стрічки деталями
```

```
Remove:   BOOL      := FALSE;           //Видалення деталей зі стрічки
END_VAR
```

VAR

```
Belt1      : "Conveyor_belt";           //Керування стрічкою 1
belt 1 Belt2 : "Conveyor_belt";           //Керування стрічкою 2
belt 2 Belt3 : "Conveyor_belt";           //Керування стрічкою 3
belt 3 Belt4 : "Conveyor_belt";           //Керування стрічкою 4
END_VAR
```

```
BEGIN
```

NETWORK 1

```
TITLE = Initializing the common signals           //Ініціалізація загальних сигналів
```

```
A   Man_start;
=   Belt1.Man_on;
=   Belt2.Man_on;
=   Belt3.Man_on;
=   Belt4.Man_on;
A   Stop;
=   Belt1.Stop;
=   Belt2.Stop;
=   Belt3.Stop;
=   Belt4.Stop;
A   Reset;
=   Belt1.Basic_state;
=   Belt2.Basic_state;
=   Belt3.Basic_state;
=   Belt4.Basic_state;
```

NETWORK 2

```
TITLE = Calling the conveyor belt controls           //Виклик блоків керування
                                                //окремими конвеєрами
```

```
CALL Belt1 (Start:= Start, Readyload := Load, End_of_belt:= Light_barrier1,
            Mfault:= "/Mfault1", Belt_mot_on := Belt_mot1_on);
```

```
A   Belt2.Readyload;
=   Belt1.Continue;
A   Belt1.Ready_rem;
=   Belt2.Start;
```

```
CALL Belt2 (End_pf_belt := Light_barrier2, Mfault := "/Mfault2",
            Belt_mot_on := Belt_mot2_on);
```

```
A   Belt3.Readyload;
=   Belt2.Continue;
A   Belt2.Ready_rem;
=   Belt3.Start;
```

```
CALL Belt3 (End_of_belt := Light__barrier3, Mfault:= "/Mfault3", Belt_mot_on :=
            Belt_mot3_on);
```

```
A   Belt4.Readyload;
=   Belt3.Continue;
A   Belt3.Ready_rem;
=   Belt4.Start;
```

```
CALL Belt4 (Continue:= Removed, Ready_rem:= Remove, End_of_belt:=
            Light_barrier4, Mfault:= "/Mfault4", Belt_mot_on:= Belt_mot4_on);
```

NETWORK 3

```
TITLE = Call for counting and monitoring //Контроль часу/режиму  
CALL Check (Set:= Start, Acknowledge:= "Acknowledge"; Light_barrier := Light  
barrier 1, Quantity:= #quantity, Tim:= #tim, Dura1:= #dura1, Dura2:=  
#dura2, Finished:= Finished, Fault:= "Fault");
```

NETWORK 4

```
TITLE = Block end  
BE  
END FUNCTION BLOCK
```

Завдання керування можуть змінюватися в різних випадках. Так, наприклад, у ручному або налагоджувальному режимах може знадобитися поштовховий запуск двигуна. В іншій ситуації може виникнути необхідність виміру ваги виробу. Можливе введення допоміжних функцій, наприклад, підрахунок циклів запуску для проведення обслуговування (очищення, змащення й ін.) транспортера при досягненні певного відпрацьованого ресурсу.

3.4 Методика виконання індивідуального завдання

Варіанти індивідуальних завдань наведені в Додатку В.

Номер варіанта відповідає порядковому номеру студента в журналі групи.

У завданні зазначено, якими функціями повинна бути доповнена базова програма керування конвеєром.

При виконанні роботи необхідно зробити наступне:

1. Модернізуйте структуру програми так, щоб вона дозволяла швидко змінювати склад функцій і розв'язуваних завдань.
2. Додайте необхідні засоби керування й індикації на пульт керування.
3. Сконфігуруйте апаратуру для реалізації програми.
4. Складіть нову таблицю глобальних змінних.
5. Розробіть ієрархію викликів функцій і функціональних блоків.
6. Доробіть програмні блоки відповідно до нових завдань.

У звіті по роботі необхідно представити:

1. Завдання (варіант).
2. Структура програми й схема викликів програмних блоків.
3. Документація по програмі, таблиця глобальних змінних.
4. Програма в електронному варіанті.

4 НАЛАГОДЖЕННЯ ПРОГРАМИ В S7-PLCSIM

Ціль роботи: освоєння методики й правил налагодження програми в додатку S7-PLCSIM.

4.1 Загальні відомості про S7-PLCSIM

S7-PLCSIM дозволяє протестувати створену програму на імітаторові програмувального логічного контролера (ПЛК), тобто на комп'ютері.

S7-PLCSIM підтримує імітацію таймерів і лічильників, входів-виходів із загальною адресуємою пам'яттю 16 Кбайт, меркерів, логічних блоків FB і функцій (FC), а також блоків даних (DB), системних функціональних блоків (SFB) і системних функцій (SFC).

PLCSIM підтримує також організаційні блоки:

- OB1 (цикл);
- OB10 - OB17 (переривання за часом дня);
- OB20 - OB23 (переривання із затримкою);
- OB30 - OB38 (циклічні переривання);
- OB40 - OB47 (апаратні переривання);
- OB70 - OB73 (помилки резервування);
- OB80 (помилка часу);
- OB82 (діагностичне переривання);
- OB100 (повний перезапуск, гаряче перезавантаження);
- OB101 (тепле перезавантаження);
- OB102 (холодне перезавантаження).

Щоб спостерігати хід процесу, потрібно створити «видимі об'єкти». При налагодженні програми можна записати ряд подій (маніпуляція областями пам'яті входу й виходу, суматорами, регістрами) і відтворити запис у режимі автоматичного тестування програми.

З використанням таблиці символів (Symbols) можна відобразити наступні типи видимих об'єктів:

Вхідна змінна: дозволяє одержати доступ до даних, збережених в області пам'яті входів (I). Значення адреси за замовчуванням (IB 0) можна змінювати.

Вихідна змінна: дозволяє одержати доступ до даних, збережених в області пам'яті виходів (Q). Значення адреси за замовчуванням – QB 0.

Меркери: дозволяє одержати доступ до даних, збережених в області меркерів (M). Початкове значення адреси – це байт MB 0.

На рисунку 4.1 показаний приклад відображення деяких змінних для програми керування конвеєром.

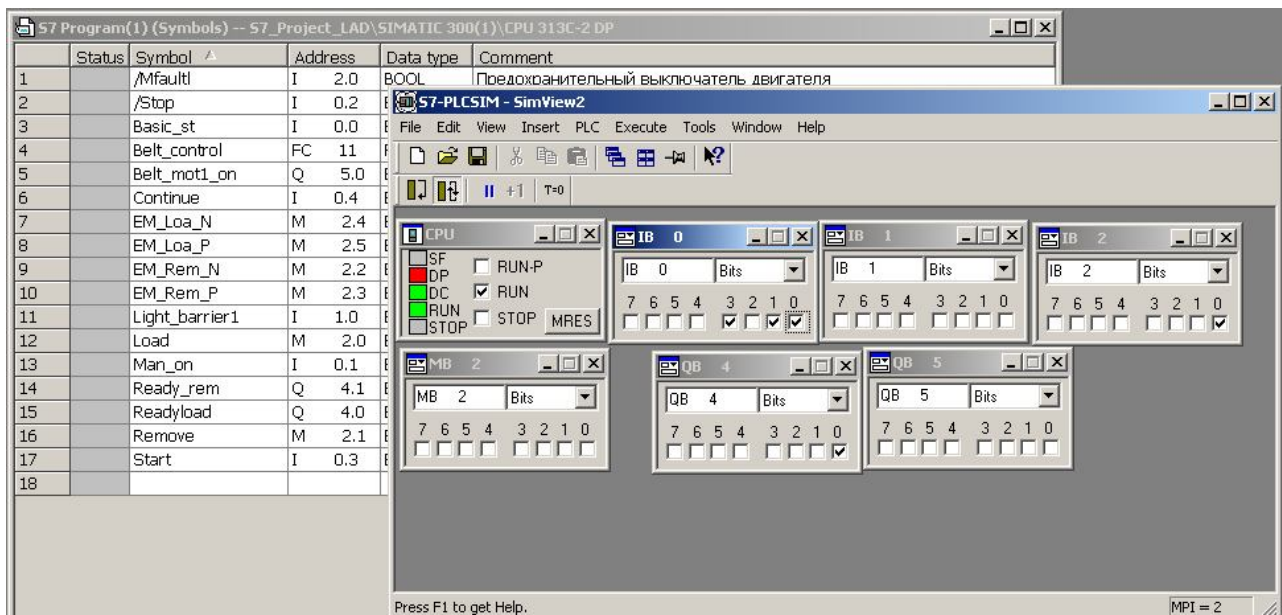


Рисунок 4.1 – Відображення "видимих об'єктів" в PLCSIM

Крім зазначених типів об'єктів PLCSIM дозволяє також відображати таймери й лічильники, використовувані програмою, а також одержати доступ до будь-якої області пам'яті в імітаторі CPU, включаючи блок даних (DB).

Наступні три видимі об'єкти активуються в меню View (Вид):

Акумулятори: дозволяють відобразити дані в різних акумуляторах в імітаторі CPU, а також слово стану й адресні регістри. Видимий об'єкт відображає чотири акумулятори CPU S7-400. Програма для CPU S7-300 використовує тільки два акумулятори.

Регістри блоків: дозволяють відобразити зміст адресних регістрів блоків даних в імітаторі CPU. Також показують номер виконуваного логічного блоку й номер попереднього логічного блоку, з номером виконуваної інструкції (лічильника адреси, або SAC).


Стеки: дозволяють відобразити збережені дані в апаратних стеках і стеці команди MCR (в імітаторі ПЛК).

Великою гідністю PLCSIM є можливість спостерігати функціонування програми не тільки по видимих об'єктах, але й безпосередньо в редакторі «LAD/STL/FBD» (рис. 4.2).

4.2 Методика роботи в S7-PLCSIM

Є кілька можливостей почати роботу в S7-PLCSIM:

1. Зі стартового меню Windows, вибравши програму S7-PLCSIM.

2. З панелі інструментів SIMATIC Manager, натиснувши кнопку Simulation On/Off .
3. Вибравши команду меню Option ⇒ Simulate Modules (Опції ⇒ Імітувати модулі).

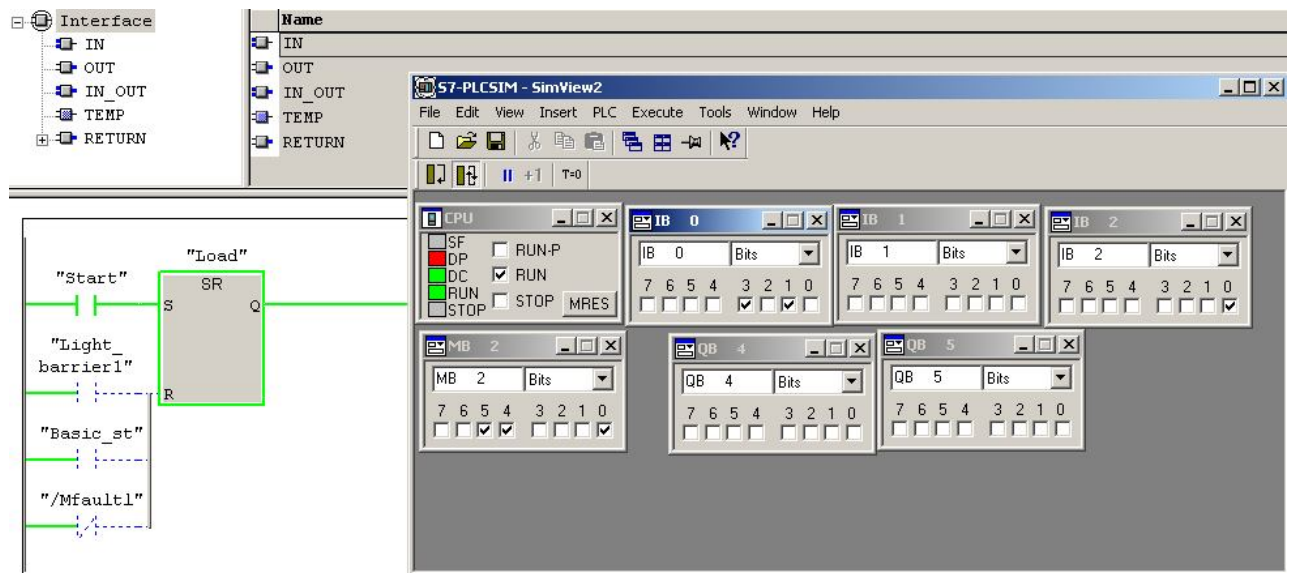



Рисунок 4.2 – Сполучення "LAD/STL/FBD" з "PLCSIM"

Після запуску S7-PLCSIM потрібно відкрити новий імітуємий ПЛК або продовжити імітацію раніше створеного ПЛК. Використовуючи команди File ⇒ Recent Simulation (Файл ⇒ Остання імітація) або File ⇒ Open PLC... (Файл ⇒ Відкрити ПЛК ...), можна вибрати файл із розширенням *.plc, у якому була збережена попередня програма.

Файли *.plc використовується для зберігання інформації про роботу, яка виконувалася процесором з видимими об'єктами, тобто на імітаторові ПЛК. У нього також записуються ті зміни, які пов'язані із присвоєнням значення деякої змінної в пам'яті.


Крім файлу *.plc в PLCSIM створюються файли *.lay, які використовуються для зберігання потрібного порядку видимих об'єктів.

S7-PLCSIM пропонує наступні можливості для виконання імітуємої програми:


1. Однократне виконання програми, коли CPU виконає один цикл. Кожний цикл складається із читання CPU периферійних входів (PI), виконання програми й запису результатів у периферійні виходи (PQ). CPU потім чекає, коли буде запущений наступний цикл або командою меню Execute ⇒ Next Scan, або натисканням кнопки .
2. Циклічний режим, коли CPU виконує один повний цикл і потім починає інший.

Для того, щоб освоїти роботу в S7-PLCSIM, виконайте наступне:


1. Відкрийте SIMATIC Manager.


2. Натисніть на  або виберіть команду меню Options ⇒ Simulate Modules (*Опції ⇒ Імітація модулів*). Цією дією відкривається додаток S7-PLCSIM з видимим об'єктом CPU (за замовчуванням адреса MPI – «2»).


3. У SIMATIC Manager відкрийте файл проекту ZEN01_09_STEP7_Zebra. У цьому проекті виділіть папку Blocks (блоки).

4. У SIMATIC Manager натисніть на  або виберіть команду меню PLC ⇒ Download (*ПЛК ⇒ Завантажити*) для завантаження блоків програми в імітатор ПЛК. На запитання «чи прагнете Ви завантажити системні дані?», виберіть No (*Ні*), якщо Ви не прагнете завантажити апаратну конфігурацію в імітатор ПЛК, або Yes (*Так*) для завантаження апаратної конфігурації.

5. У додатку S7-PLCSIM створіть наступні «видимі об'єкти» для спостереження інформації в імітаторові ПЛК:

- Натисніть на  або виберіть команду меню Insert ⇒ Input Variable (*Вставити ⇒ Вхідні змінні*). З'являється видимий об'єкт IB 0 (вхідний байт 0).

- Натисніть на  або виберіть команду меню Insert ⇒ Output Variable (*Вставити ⇒ Вихідні змінні*) для того, щоб відзначити другий видимий об'єкт QB 0 (вихідний байт 0).


- Натисніть на  або виберіть команду меню Insert ⇒ Timer (*Вставити ⇒ Таймер*) три рази для того, щоб позначити три видимі об'єкти. Наберіть 2, 3 і 4 (для таймерів T 2, T 3 і T 4) у відповідних текстових боксах, натискаючи клавішу Enter (*Уведення*) після кожного введення.

6. Виберіть в S7-PLCSIM меню PLC і переконайтеся, що пункт Power On (*Живлення включене*) відзначений значком (√).

7. Виберіть у меню команду Execute ⇒ Scan Mode (*Виконати ⇒ Режим циклів*) і переконайтеся, що Continuous Scan (*Циклічна робота*) відзначена значком (√).

8. Перемкніть імітатор CPU в режим роботи, відзначивши бокс вибору RUN або RUN-P.

9. Відзначте біт 0 в IB 0, щоб імітувати сигнал на вході I 0.0 і подивіться роботу таймерів і зміну вихідних сигналів QB 0.

Для спостереження програми в «LAD/STL/FBD» необхідно в SIMATIC Manager вибрати в дереві проекту блоки (Blocks) і в меню PLC подати команду Download. При цьому імітатор повинен перебувати в стані STOP. Потім в PLCSIM перемкнутися в режим RUN або RUN-P. Після цього в редакторі LAD/STL/FBD установити режим Online, натиснувши на , а потім у меню Debug вибрати команду Monitor. Додаток «LAD/STL/FBD» буде показувати програму, яка виконується імітатором ПЛК.

Робочі режими CPU

Режим RUN-P. CPU виконує програму й дозволяє змінювати програму і її параметри. У цьому режимі можна використовувати видимі об'єкти для зміни будь-яких даних, використовуваних програмою.

Режим RUN. CPU виконує програму із опитуванням входів і відновленням виходів. Однак у цьому режимі не можна завантажити іншу програму або змінити її параметри. Режим дозволяє використовувати видимі об'єкти, створені за допомогою S7-PLCSIM, а також змінювати будь-які дані, використовувані програмою.

Режим STOP. CPU не виконує програму. На відміну від режиму STOP реального CPU виходи не встановлюються в безпечні значення, але залишаються в стані, у якому вони були, коли CPU перейшов в STOP. У режимі STOP можна завантажити програму в CPU. Перехід з режиму STOP в RUN викликає виконання програми, починаючи з першої команди.

Індикатори CPU

Видимий об'єкт CPU відтворює індикатори реального CPU:

- SF (системна несправність) сигналізує, що CPU зустрів системну помилку, що привела до зміни робочого режиму.
- DP (розподілена периферія або вилучені введення/виводи) показує стан зв'язку з розподіленим (вилученим) входом/виходом.
- DC (забезпечення живленням) показує включення або виключення живлення CPU.
- RUN показує, що CPU перебуває в режимі RUN.
- STOP показує, що CPU перебуває в режимі STOP.

Області пам'яті

В PLCSIM можна одержати доступ до певних областей пам'яті, які виконують спеціальні функції:

- PI (периферійний вхід): забезпечує прямий доступ до вхідних модулів.
- I (вхід): забезпечує доступ до області відображення входів (ці величини обновляються CPU на початку кожного циклу).
- PQ (периферійний вихід) забезпечує прямий доступ до вихідних модулів (ці значення обновляються CPU наприкінці кожного циклу).
- Q (вихід): забезпечує доступ до області відображення виходів.
- M (меркери): забезпечує зберігання даних, використовуваних усередині програми.
- T (таймер): забезпечує зберігання таймерів.
- C (лічильник): забезпечує зберігання лічильників.

Крім того можна одержати доступ до даних, що зберігаються в блоках даних (DB).

Конфігурування входів/виходів

Для того щоб імітувати ОВ переривань, потрібно завантажити апаратну конфігурацію, яка містить адреси входів/виходів.

CPU S7-315-2DP, S7-316-2DP і S7-318-2 завантажують конфігурацію входів/виходів. Усі інші CPU S7-300 автоматично конфігурують входи/виходи, що збігаються з фізичними входами/виходами, установленими в стійку.

Якщо використовуються CPU S7-400 із PROFIBUS-DP, то завантажити конфігурацію входів/виходів і використовувати її для імітації переривання ОВ в S7-PLCSIM не вдасться. Однак можна скопіювати конфігурацію входів/виходів у другий проект і замінити CPU на ту модель, наприклад, 416-DP, яка явно підтримує DP.

Збереження імітатора ПЛК

Поточний стан в імітаторі ПЛК можна зберегти в такий спосіб:

- Командою меню File ⇒ Save PLC (*Файл ⇒ Зберегти ПЛК*) для збереження конфігурації ПЛК у поточному файлі.
- Командою меню File ⇒ Save PLC As... (*Файл ⇒ Зберегти ПЛК як ...*) для збереження конфігурації ПЛК у новому файлі.

Для того, щоб зберегти конфігурацію видимих об'єктів, використовуйте команду меню File ⇒ Save Layout (*Файл ⇒ Зберегти компоновання*).

При збереженні ПЛК зберігаються програма, апаратна конфігурація, опції керування (циклічна робота, однократне виконання), стан входів/виходів, значення таймерів (Т-пам'ять), символні адреси, стан живлення (включене/виключене).

Коли відкривається імітатор ПЛК, незалежно від того, чи новий це імітатор, чи збережений, він перебуває в режимі STOP.

Для того, щоб показати символні адреси, використовуйте команду меню Tools ⇒ Options ⇒ Show Symbols (*Інструменти ⇒ Опції ⇒ Показати символіку*).

Закінчення роботи імітатора

Після збереження будь-якого імітатора ПЛК або конфігурації, слід проробити наступні кроки для виходу з S7-PLCSIM:

1. Закрийте всі STEP 7 програми, залучені в імітацію.
2. Виберіть команду меню **File** ⇒ **Exit** (*Файл ⇒ Вийти*).

4.3 Методика виконання роботи й зміст звіту

Завданням даної роботи є тестування й налагодження програми, створеної в попередній роботі.

При виконанні роботи необхідно зробити наступне:

1. Відкрийте Simatic Manager.
2. В Simatic Manager відкрийте Ваш проект, виділіть папку "Blocks", а потім запустіть PLCSIM і завантажте блоки програми в імітатор ПЛК.
3. Використовуючи таблицю символів і розділи оголошення змінних у блоках, створіть в PLCSIM необхідні для тестування видимі об'єкти.
4. Відкрийте тестуємий блок програми в редакторі LAD/STL/FBD і встановіть режим Online.
5. Переведіть імітатор ПЛК у режим RUN-P.
6. Установіть за допомогою видимих об'єктів необхідні параметри на вході блоку й, використовуючи спостереження, проаналізуйте поведінку програми.
7. Якщо поведінка не відповідає необхідній, виправте помилки програмування.
8. Збережіть файли конфігурації ПЛК і компонування видимих об'єктів.

Звіт по роботі повинен містити опис завдань тестування й результатів тестування, представлених скріншотами (знімки екрана).

При захисті звіту необхідно пояснити прийнятну методику тестування програми, а також обґрунтувати достатність тестових операцій.

Захист роботи повинен супроводжуватися демонстрацією роботи програми в імітаторові.

5 СТВОРЕННЯ ПРОГРАМИ МОВОЮ S7-HIGRAPH

Ціль роботи: освоєння інтерфейсу й придбання навичок створення первинників («исходники» на російській мові) у редакторі S7-HiGraph – інструментальному додатку програмної системи STEP 7.

5.1 Принцип програмування мовою S7-HiGraph

Графічна нотація широко використовується для опису поведінки автоматів, які здійснюють логічне керування встаткуванням. Алгоритми керування часто представляються блок-схемами, перемикальними схемами, мережами Петрі й т.п. Мова програмування S7-HiGraph дозволяє розширити функціональну область середовища програмування STEP 7 шляхом застосування графічного методу на основі використання графів станів.

Для застосування цієї мови об'єкт автоматизації розділяється на функціональні матеріальні одиниці. Поведінка кожної функціональної одиниці описується графом станів. Для організації взаємодії графів станів створюється координуючий граф станів (граф-диспетчер). Усі процедури процесу створення графів станів здійснюються в редакторі S7-HiGraph.

HiGraph-програма структурована в такий спосіб:

- У графах станів визначаються дії, які можуть бути зроблені при вході в стан, під час стану і при виході зі стану.
- Керування переходом (транзакцією) з одного стану в інший здійснюється розв'язною умовою із заданим рівнем пріоритету.
- Дії в станах і умови в транзакціях описуються мовою програмування STEP 7 STL.
- Для забезпечення взаємодії графів станів з координуючим графом станів програмуються повідомлення – вихідні в станах і вхідні в транзакціях.
- Графи станів разом з координуючим графом (первинники) вставляються в груповий граф. У груповому графові призначаються поточні параметри всім змінним і повідомленням.
- Груповий граф компілюється зі створенням функції керування (FC) і блоку даних поточних значень параметрів (DB), які розміщуються в контейнері Blocks S7-програми.

Увесь алгоритм керування може бути задокументований у графічній і текстовій формі.

Функція HiGraph FC повинна викликатися із циклічно працюючого блоку OB1.

Структура готової програми показана на рисунку 5.1.

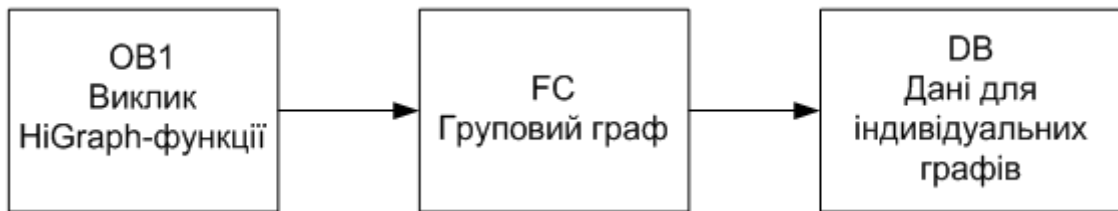


Рисунок 5.1 – Структура HiGraph-програми

Редактор HiGraph забезпечує наступні функції програмування:

- Програмування станів і транзакцій мовою STL.
- Програмування викликів FC-функції HiGraph і логічних блоків STEP 7 (FC, SFC, FB, SFB), створених із застосуванням мов STL, LAD, FBD, а також SCL-команд.
- Програмування часу очікування завершення дії й контрольного часу перебування в стані.
- Тестування функціональних блоків з визначенням активного стану, попереднього стану й останньої транзакції, а також з виставою інформації щодо команд у станах і транзакціях.
- Виявлення помилок процесу, блокувань за часом і аварійних ситуацій з виводом інформації на пристрій зв'язку з оператором.

Графи станів і групові граfi зберігаються в SIMATIC Manager у контейнері "Source files" (первинники), а скомпільовані групові граfi у вигляді функції FC, блоку даних DB і додаткових блоків – у папці Blocks.

Редактор HiGraph забезпечує наступні опції програмування:

- Вставку будь-якої кількості графів стану в груповий граф.
- Одночасне редагування декількох групових графів.
- Програмування умов у транзакціях.
- Програмування дій у станах, причому дії характеризуються подіями на вході (E), діями на виході (X) і циклічними діями (C и C-).
- Використання для програмування всього спектра STL-команд, перелік команд наведений у додатку.
- Уведення контрольного часу й часу очікування у формі змінних або констант.
- Перемикання між символною й абсолютною виставою адрес.

Процес створення програми містить у собі наступні кроки:

1. Створення проекту програми в Simatic Manager.
2. Вставка графа станів у програму.
3. Визначення сигналів, необхідних для керування.
4. Програмування станів.
5. Програмування транзакцій.

6. Програмування постійних інструкцій.
7. Програмування операційних режимів (автоматичний, ручний).
8. Створення координуючого графа.
9. Створення групового графа.
10. Установка послідовності виконання.
11. Призначення фактичних параметрів.
12. Програмування повідомлень.
13. Компіляція первинників і створення блоків програми.
14. Завантаження програми в контролер.
15. Налаштування програми в інтерактивному режимі.

У першій роботі практикуму, розрахованій на 4 години занять, необхідно виконати кроки 1-8, у другій роботі, такої ж тривалості, завершити створення програми (кроки 9-15).

1.2 Приклад виділення графів станів у завданні керування

Розглянемо приклад, наведений у Посібнику користувача «S7-Higraph V5.3 Programming State Graphs. Programming and Operating Manual». У прикладі створюється проста програма для свердлильного верстата, показаного на рисунку 5.2.

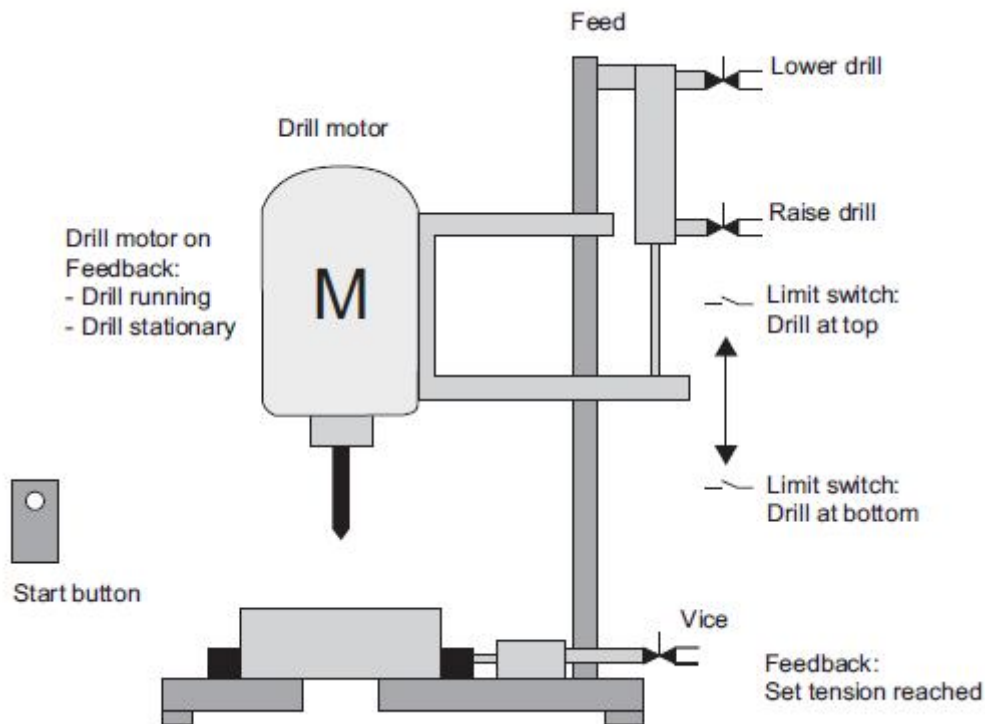


Рисунок 5.2 – Функціональні елементи свердлильного верстата

Свердлильний верстат містить гідравлічний затискач заготовки (Vice, лещата), керований за допомогою золотника, електропривод обертання свердла (Drill motor) і гідравлічний привід подачі свердлильної головки (Feed) із

золотниками керування підйомом (Raise drill) і опусканням (Lower drill). Контроль процесу здійснюється кінцевими вимикачами переміщення свердлильної головки (Limit switch), датчиком швидкості обертання вала двигуна (Motor Running) і тензодатчиком (Tension Reached), який сигналізує про досягнення необхідної сили затискача заготовки. Пуск процесу свердління здійснюється кнопкою Start button.

Початковий стан верстата визначений у такий спосіб:

- Мотор привода свердла зупинений.
- Свердлильна головка в крайньому верхньому положенні.
- Заготовка встановлена в затискному пристосуванні, не затиснута.

На рисунку 5.3 показана функціональна діаграма процесу свердління, що складається із 8 станів.

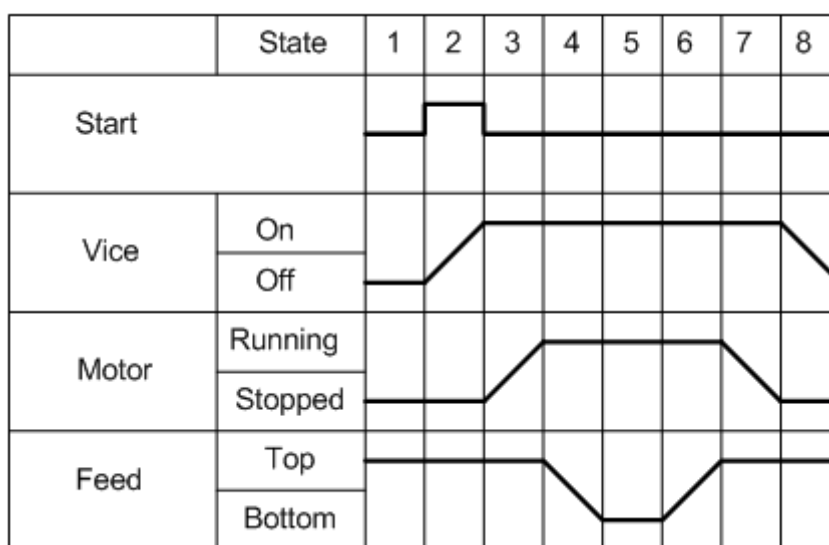


Рисунок 5.3 – Функціональна діаграма процесу свердління

Процес свердління в автоматичному режимі починається із включення верстата пусковою кнопкою Start button і складається з наступних операцій:

1. Затискання заготовки поки не буде досягнутий тиск фіксації.
2. Пуск двигуна обертання свердла.
3. Подача свердла вниз, поки не досягнута крайня нижня точка.
4. Витримка в нижньому положенні.
5. Подача свердла нагору, поки не досягнута крайня верхня точка.
6. Зупинка двигуна обертання свердла.
7. Розтискання заготовки (заготовка видаляється оператором).

Виконавчі пристрої свердлильного верстата управляються через виходи модуля цифрового виводу з адресами Q 0.0 - Q 0.7. Вхідні сигнали подаються на модуль введення з адресами входів I 0.0 - I 0.7.

Призначення вхідних і вихідних сигналів наведено в таблиці 5.1.

Таблиця 5.1

Адреса		Опис
Символьна	Абсол.	
Drill_Motor_Running	I 0.0	Свердло обертається із заданою швидкістю
Drill_Motor_Stopped	I 0.1	Двигун привода свердла зупинений
Drill_at_Bottom	I 0.2	Свердлильна головка в нижньому положенні
Drill_at_Top	I 0.3	Свердлильна головка у верхньому положенні
Tension_Reached	I 0.4	Заготовка затиснута (тиск досягнутий)
Start_Button	I 0.7	Сигнал пускової кнопки
Drill_Motor_On	Q 0.0	Включити двигун привода свердла
Lower_Drill	Q 0.1	Включити подачу свердла вниз
Raise_Drill	Q 0.2	Включити подачу свердла нагору
Clamp_Workpiece	Q 0.3	Затиснути заготовку

Із опису завдання випливає, що процес свердління заготовки здійснюється за допомогою трьох функціональних одиниць – пристрою затискача деталі (Vice), привода обертання свердла (Motor) і пристрою подачі свердлильної головки (Feed). Для кожної функціональної одиниці потрібен один граф станів. Для координації роботи цих функціональних одиниць потрібен ще один граф станів – свердління (Drilling).

Порядок створення графа станів можна розглянути на прикладі привода подачі «Feed».

На рисунку 5.4 показаний функціональний блок гідроциліндра подачі з діаграмами сигналів керування й зворотного зв'язку. Він містить два електромагнітні клапани (Up і Down) і два кінцеві вимикачі (Top і Bottom).

Як видно з діаграми, процес керування гідроциліндром складається із чотирьох станів – 1, 2, 3, 4.

На діаграмі не показаний стан 0, який повинен бути в кожному графові. Він призначений для перевірки поточного положення функціонального блоку і його перекладу при необхідності у стан готовності (свердлильна головка перебуває у верхньому положенні). Вивід свердла у верхнє положення повинен здійснюватись по закінченню кожного циклу, тобто в стані 4. Якщо з якоїсь причини в момент включення верстата свердло перебуває в іншому положенні, то перехід із цього стану (стан 0) повинен бути зроблений саме в стан 4, щоб забезпечити переміщення свердла нагору. З обліком цього складена послідовність зміни станів графа «Feed» (рис. 5.5).

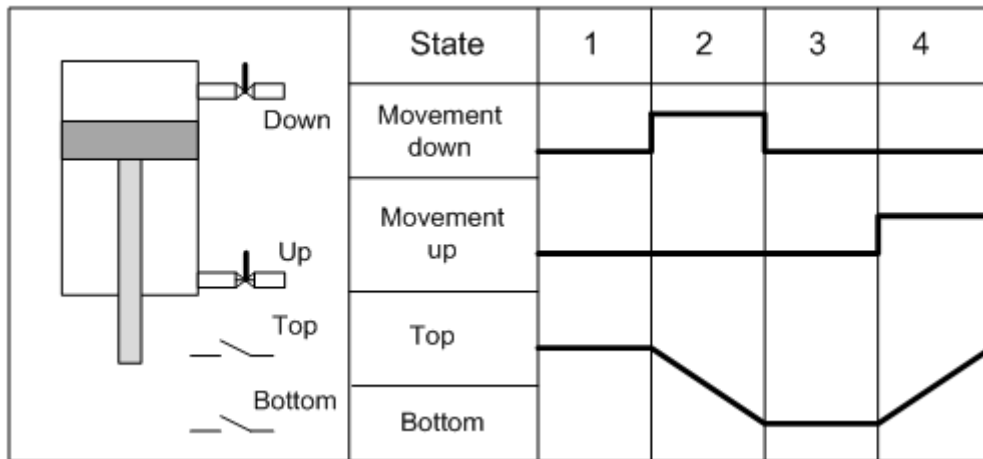


Рисунок 5.4 – Функціональний блок гідроциліндра подачі свердла



Рисунок 5.5 – Послідовність виконання графа станів «Feed»

Передбачається, що клапани з електромагнітним керуванням повинні використовуватися тільки для фази руху й гідроциліндр залишається у верхньому кінцевому положенні при знятті сигналів керування.

Приклад програми «Drilling machine» (Zen03_02_Higraph_Drillmac) перебуває в папці Sample projects (Типові проекти).

5.3 Послідовність створення графа станів в HiGraph

Розглянемо деталі створення графа станів у кожному кроці.

Крок 1. Створення проекту програми в Simatic Manager.

Для створення нової програми необхідно відкрити Simatic Manager, і у вікні «New Project» увести ім'я нового проекту й натиснути ОК.

У лівій панелі центрального вікна Simatic Manager установіть курсор на імені проекту й правою кнопкою миші викличте контекстне меню. У цьому меню виберіть команди Insert New Object ► SIMATIC 300 Station. Далі у вікні HW-config під станцію SIMATIC 300 потрібно вставити рейку RACK 300 (Rail), блок живлення й процесорний модуль CPU315-2 PN/DP. Процес компонування елементів проекту закінчується вставкою S7 Program, яка буде відображена в правій панелі.

Крок 2. Вставка графа станів у програму.

Для вставки графа станів потрібно подвійним клацанням лівої кнопки миші розкрийте структуру S7 Program, виберіть контейнер «Sources» і правою кнопкою виберіть контекстне меню, у якому виберіть Insert New Object ► State graph. При цьому в правій панелі з'явиться об'єкт «State graph(1)», якому тут же слід привласнити ім'я, наприклад, «Motor». Після подвійного клацання по піктограмі «Motor» тільки що створений початковий граф станів буде відображений в основному вікні редактора HiGraph (рис. 5.6).

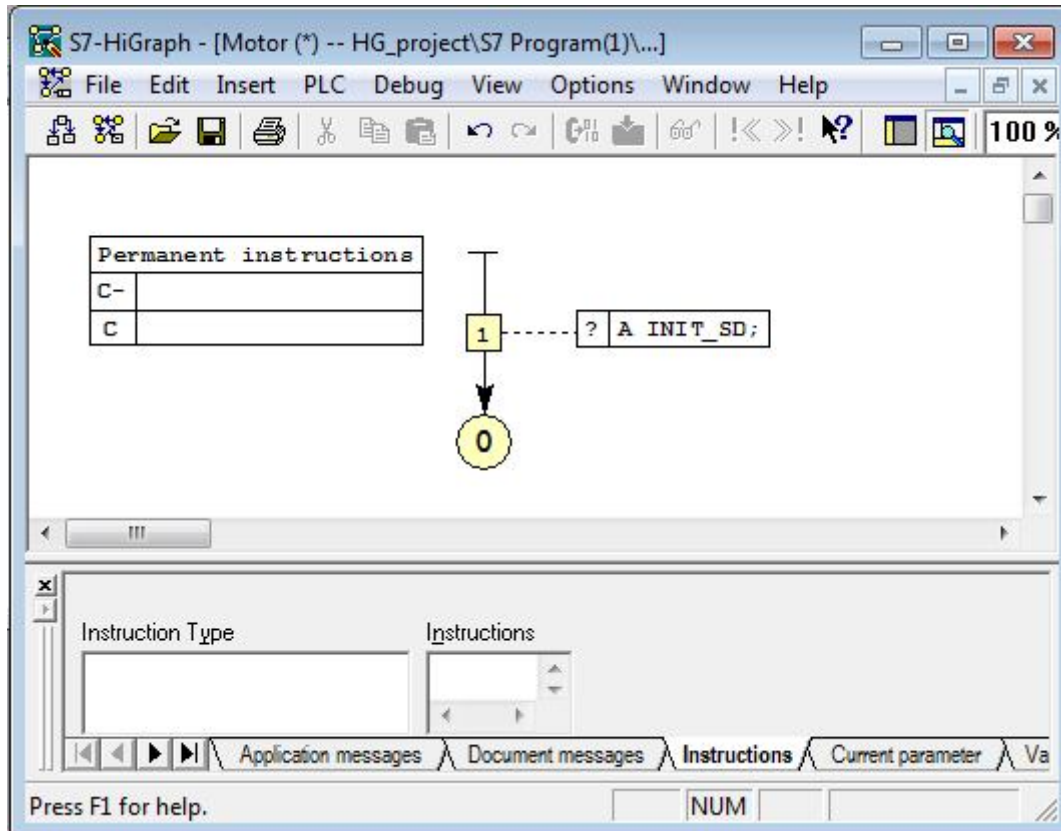


Рисунок 5.6 – Вікно редактора з початковим графом станів

Під вікном редактора показано відкрите вікно інструкцій (Instructions), у якому можна ввести команди для програмування станів і транзакцій. За допомогою кнопок унизу вікна можна також відкрити вікна оголошення змінних (Variables) і повідомлень (Application messages і Document messages).

Крок 3. Визначення сигналів, необхідних для керування.

Для відображення розділу змінних виберіть в меню View опцію Details. При цьому відкриється ліва панель Environment (навколишнє середовище), у якій відображаються елементи програми, згруповані в розділи. Для перегляду змінних слід розкрити розділ Interface. У нижній частині вікна редактора перебуває вікно редагування змінних, яке відкривається кнопкою Variables. Це вікно являє собою таблицю із чотирма колонками – ім'я, тип даних, коментар і початкове значення. При цьому початкове значення не редагується, а встановлюється системою. Вид вікна редагування після зроблених у цьому кроці налаштувань показано на рисунку 5.7.

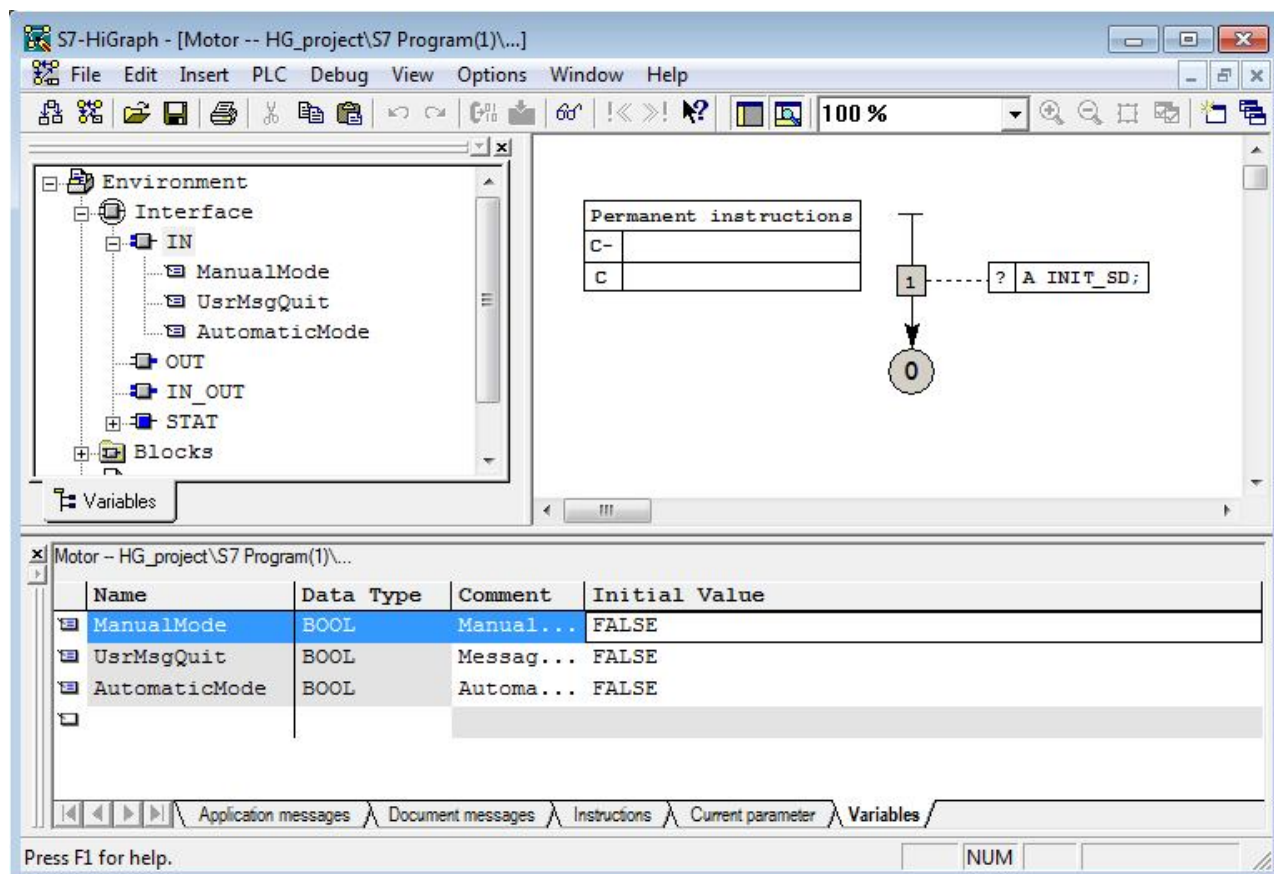


Рисунок 5.7 – Вид вікна редагування на кроці 3

Розділ змінних містить наступні значення:

- *Вхідні змінні IN.* При створенні графа редактор автоматично вводить у список вхідних змінних три змінні – AutomaticMode, ManualMode і UsrMsgQuit, видаляти які не можна. При цьому вхідна змінна AutomaticMode при значенні «1» дозволяє обробку транзакцій тільки з атрибутом «Auto» і забороняє обробку транзакцій з атрибутом «Manual». Аналогічно, змінна

ManualMode при значенні «1» дозволяє обробку транзакцій тільки з атрибутом «Manual». Змінна UsrMsgQuit служить для підтвердження помилки або повідомлення.

- *Вихідні змінні OUT.* У список вихідних змінних уводяться імена вихідних параметрів графа станів.

- *Вхідні/вихідні змінні IN_OUT.* У цей список включаються змінні для обміну повідомленнями між графами станів.

- *Змінні STAT.* Для спрощення процесу програмування редактор HiGraph автоматично встановлює в цьому списку 15 необхідних для роботи змінних, у тому числі INIT_SD, яка забезпечує запуск програми. Значення змінних представлені в табл. 5.2.

Таблиця 5.2 – Значення змінних розділу STAT

Ім'я змінної	Значення	Тип даних	Визначається	
			Користувачем	Системою
WT_Expired	Витікання часу очікування	BOOL		Так
WT_Valid	Час очікування активно	BOOL		Так
WT_Stop	Зупинка часу очікування	BOOL	Так	
WT_CurrValue	Збереження часу очікування	DWORD		Так
UsrMsgSend	Повідомлення стану активно	BOOL		Так
UsrMsgStat	Для внутрішнього використання	WORD		
ST_ExpiredPrev	Витікання контрольного часу попереднього стану	BOOL		Так
ST_Expired	Витікання контрольного часу	BOOL		Так
ST_Valid	Контрольний час активний	BOOL		Так
ST_Stop	Зупинка контрольного часу	BOOL	Так	
ST_CurrValue	Збереження контрольного часу	DWORD		Так
INIT_SD	Параметр запуску	BOOL	Так	
CurrentState	Номер поточного стану	WORD		Так
PreviousState	Номер попереднього стану	WORD		Так
StateChange	Зміна стану	BOOL		Так

При оголошенні змінних дозволяється наступне.

- ✓ В імені змінної дозволяються текстові символи й символи підкреслення, причому символ підкреслення може стояти на початку імені, але не повинен стояти наприкінці імені.
- ✓ При оголошенні типу даних редактор пред'являє на вибір BOOL, INT, WORD, CHAR, String, TIME і т.д.
- ✓ Для повідомлень, які оголошуються в розділі IN_OUT, у вікні оголошень відкривається колонка Message Type (тип повідомлення).
- ✓ У розділі коментаря допускається будь-яка вистава тексту.

У вікні оголошення змінних не передбачене введення адреси. Адреса змінної буде визначена при заповненні таблиці ідентифікаторів Symbols, яка перебуває в папці S7 Program додатка Simatic Manager.

Відразу після того, як новий граф стану буде створений, змінні Currentstate, Previousstate і Statechange деактивуються.

Для активізації цих змінних необхідно:

1. Виділити змінну у вікні оголошення змінних і в контекстному меню вибрати команду Object Properties.
2. У діалоговому вікні перейти на вкладку "Attributes" і призначити значення "true" на атрибут "S7_active", як показано на рисунку 5.8.

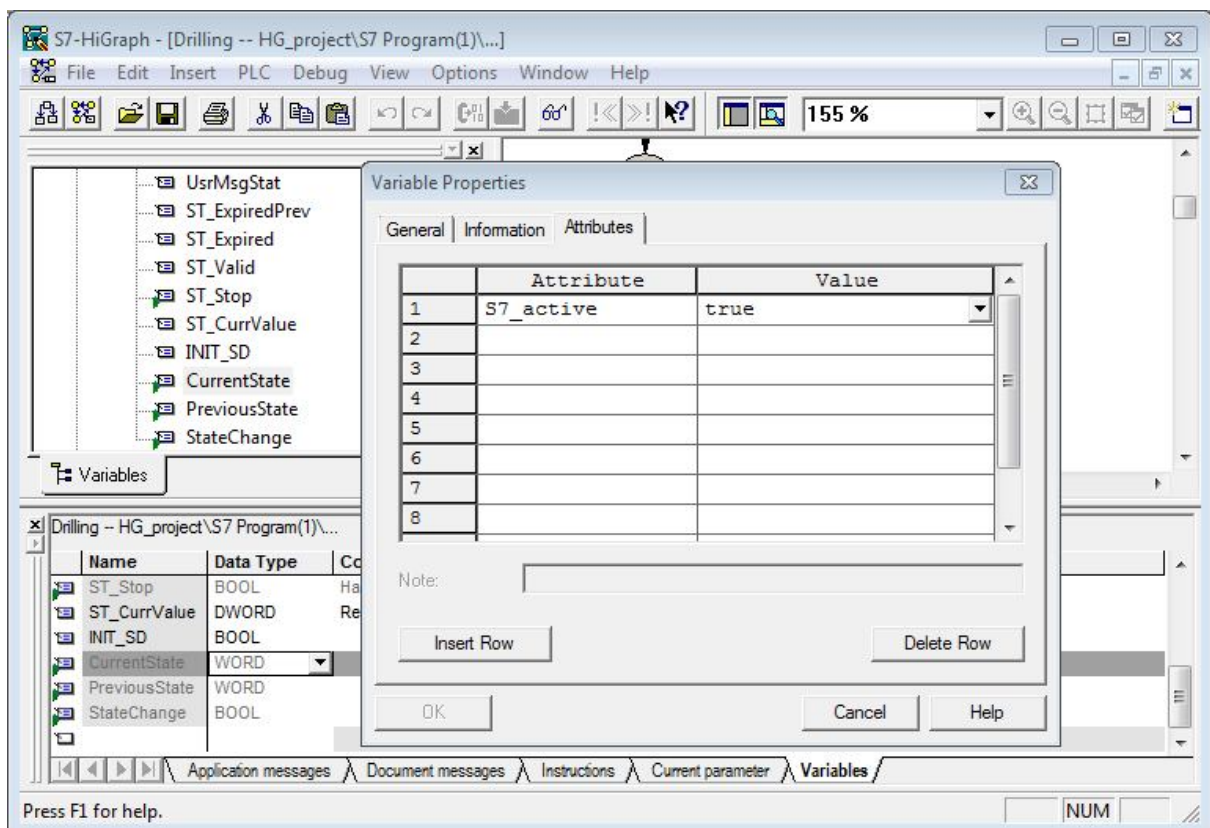


Рисунок 5.8 – Приклад активування змінної Currentstate

Крок 4. Програмування станів.

Програмування станів містить у собі:

- присвоєння імені стану (не обов'язково);
- уведення команд;
- визначення часу очікування й контрольного часу (не обов'язково);
- вставка наступного стану.

Вставка стану проводиться командою контекстного меню Insert State. Стани нумеруються в порядку, у якому вони вводяться. Присвоєння імені стану проводиться у вікні Object Propertis, виклик якого можливий після виділення стану правою кнопкою миші. У цьому ж вікні можна змінити номер стану.

Для введення команд потрібно відкрити вікно Instructions, двічі клацнувши по стану. При цьому у вікні Instructions (рис. 5.9) відобразиться список типів інструкцій. Типи інструкцій, які вводяться в стан графа, представлено в таблиці 5.3.

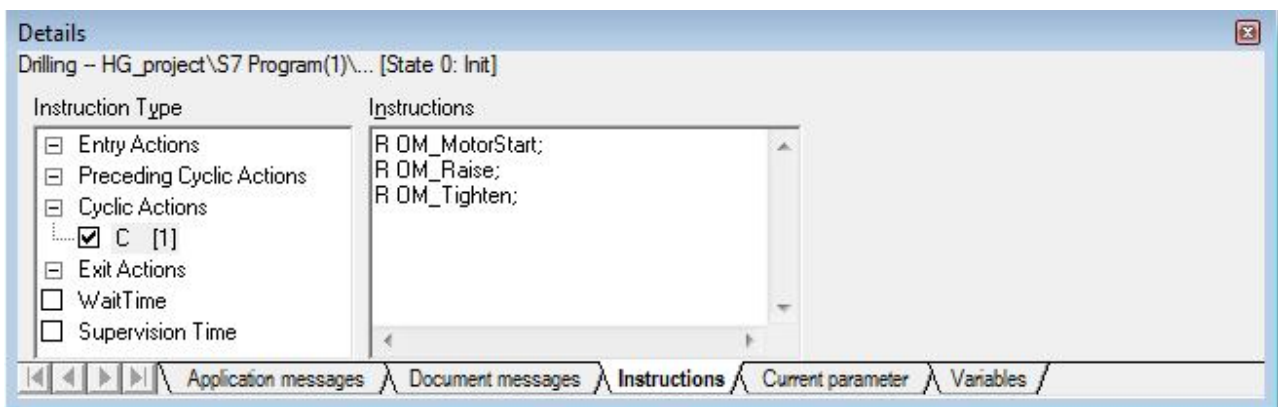


Рисунок 5.9 – Вид вікна введення команд Instructions при програмуванні стану 0

Таблиця 5.3 – Типи інструкцій, які вводяться в стані

Тип команди (інструкції)	Ідентифікатор	Опис команди
Дія входу	Е	Дія, яка виконується одного разу тільки при вході в стан
Попередня циклічна дія	С-	Дія, яка містить певні умови й виконується перед перевіркою транзакції
Циклічна дія	С	Дія, яка виконується після перевірки транзакції
Дія виходу	Х	Дія, яка виконується одного разу тільки при виході зі стану

Уведення інструкції здійснюється в правому полі, яке стає активним після виділення типу інструкції й вибору в контекстному меню єдиної команди Insert.

При введенні STL-команди можна використовувати символні й формальні параметри. Уведення повинне закінчуватися крапкою з комою. Результат програмування стану 0 показаний на рисунку 5.10.

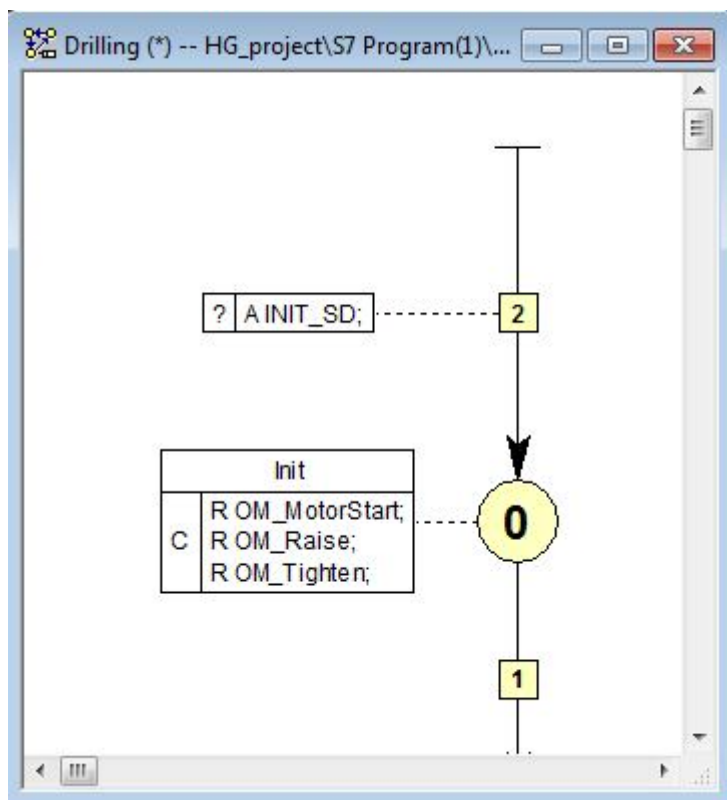


Рисунок 5.10 – Вид вікна після програмування стану 0

При програмуванні стану можна визначити, чи повинен контролер залишатися в стані якийсь час до того, як буде перевірена наступна транзакція. Установка часу очікування проводиться при виборі команди "Wait Time". При цьому в правому вікні за замовчуванням встановлюється час T#500 ms, який може бути змінений.

Якщо необхідно задати час, протягом якого процес може перебувати в стані, то слід вибрати команду "Supervision Time". Установлений за замовчуванням контрольний час 500 ms можна змінити. Якщо фактичний час перебування перевищить заданий контрольний час, то попередньо визначена змінна "ST_Expired" установиться в стан «1». При цьому в діагностичний буфер CPU буде видане повідомлення про помилку.

Для діагностики процесу виконання програми в стані можна призначити дві характеристики – помилка (функція F) і повідомлення (функція M). Ці призначення проводяться у вікні Object Properties. У цім же вікні можна призначити коментар для стану.

Крок 5. Програмування транзакцій.

Транзакція містить розв'язну умову для переходу від одного стану до іншого. Стану може бути призначено одну або кілька транзакцій, що виходять із стану. Якщо виконуються умови більше, чим однієї транзакції, то перемикавання відбудеться по транзакції з найвищим пріоритетом (1).

У мові HiGraph використовуються три типи транзакцій (рис. 5.11) – стандартна, довільна й транзакція повернення.

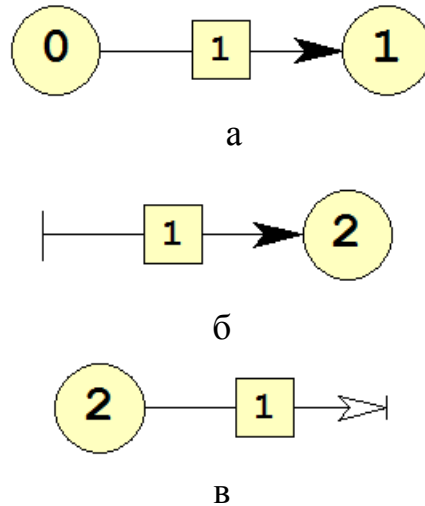


Рисунок 5.11 – Типи транзакцій

Стандартна транзакція (рис. 5.11,а) здійснює перехід зі стартового стану в наступний стан.

Довільна транзакція (рис. 5.11,б) іде від будь-яких станів до цільового стану. Вона має більш високий пріоритет, чим інші типи транзакцій і обробляється безупинно незалежно від поточного стану графа стану. Такі транзакції використовуються для постійного контролю важливих умов з високим пріоритетом. Якщо в довільній транзакції запрограмована контролююча функція виконується, то здійснюється перехід на гілку процесу із цільовим станом.

Транзакція повернення (рис. 5.11,в) іде від поточного стану до попереднього активного стану.

Вставка транзакції проводиться командою контекстного меню Insert Transition. Тип транзакції залежить від позиції кінця транзакції.

При програмуванні транзакцій виконуються наступні кроки:

- визначення пріоритету транзакції (не обов'язково);
- введення умов (обов'язково);
- введення дій транзакцій (не обов'язково);
- призначення імені транзакції (не обов'язково);
- введення коментарів (не обов'язково);
- установка часу очікування (не обов'язково);

Якщо одному стану призначено кілька транзакцій, то редактор автоматично призначає цим транзакціям різні пріоритети. Бажаний рівень пріоритету можна встановити у вікні Object Properties.

Умови (Conditions) у транзакціях записуються з ідентифікатором «?» (знак питання). Умови програмуються у форматі команд мови STL.

Дії в транзакціях (Transition actions) програмуються з ідентифікатором «!» (знак вигуку) теж у форматі команд мови STL. Ці команди виконуються однократно, коли транзакція перемикає стан.

Для введення команд необхідно двічі клацнути по транзакції, щоб відкрити вікно редагування. Далі в лівій частині вікна редагування вибрати зі списку Conditions або Transition actions і ввести STL-команду в правій області вікна. Уведені команди відображаються у вікні графа станів, як таблиця, «прикріплена» до транзакції.

Слід урахувати, що обробка команд починається з результатом логічної операції $RLO = 1$.

Крок 6. Програмування постійних інструкцій.

Постійні команди виконуються один раз у цикл, незалежно від поточного стану. У постійних командах можна програмувати наступні основні процеси:

- Обчислення змінних процесу.
- Реєстрація й обробка подій, на які процес повинен завжди реагувати, незалежно від поточного стану, наприклад, контроль захисту й блокувань.

Для редагування доступні наступні типи постійних команд:

- Циклічні дії (Preceding Cyclic Actions), які завжди виконуються на початку циклу (ідентифікатор C-).
- Циклічні дії (Cyclic Actions), які завжди виконуються наприкінці циклу (ідентифікатор C).

Щоб програмувати ці дії, потрібно клацнути по таблиці команд із заголовком "Permanent Instructions". При цьому відкриється вікно редагування, у лівій області якого необхідно вибрати тип інструкції, а в правій увести команду STL. Додавання команд проводиться так: у лівій області потрібно виділити тип інструкції, потім натиснути праву кнопку й вибрати єдину команду Insert. Після введення всіх команд вони будуть відображатися у вікні графа стану, як таблиця.

Крок 7. Програмування операційних режимів.

Якщо в транзакції запрограмувати режим Manual (ідентифікатор M), то транзакція буде перемикатися тільки в ручному режимі, а якщо запрограмувати режим Auto (ідентифікатор A), то транзакція буде перемикатися тільки в автоматичному режимі. Необхідна особливість режиму встановлюється у вікні Object Properties. Після установки режиму транзакція зафарбовується в

рожевий (Auto) або блакитний (Manual) колір і забезпечується відповідним ідентифікатором (A або M).

Крок 8. Створення координуючого графа.

З функціональної діаграми, представленої на рисунку 5.3, видно, що процес свердління складається з 8 станів.

Стан 1 характеризує початкову позицію (ініціалізацію) – лещата розціплені, свердло знаходиться у верхньому положенні й мотор виключений. У цій позиції можлива установка й зняття заготовки.

Перехід у стан 2 здійснюється пусковою кнопкою (Start_Button). Коли процес затискача закінчиться й спрацює тензодатчик зусилля затискача (Tension_Reached), транзакція перемкне процес у стан 3. У цьому стані запускається мотор обертання свердла й по досягненню заданої швидкості (сигнал Drill_Motor_running) відбувається перехід у стан 4 – включення приводу подачі свердла.

Подача проводиться до моменту спрацьовування кінцевого вимикача в крайньому нижньому положенні свердлильної головки (стан 5). У цьому положенні свердло повинне обертатися без подачі якийсь час для зменшення пружних деформацій від осьової подачі свердла й потім автомат повинен перейти в стан 6, де реалізується команда руху свердлильної головки нагору. По закінченню процесу, коли спрацює кінцевий вимикач крайнього верхнього положення головки, автомат перейде в стан 7. Тут відбувається вимикання мотора й після його зупинки (сигнал з датчика швидкості Drill_Motor_stopped) перемикає в стан 8, у якому проводиться розтискання лещат.

На рисунку 5.12 показаний координуючий граф станів Drilling, який відповідає функціональній діаграмі процесу свердління.

5.4 Вимоги до звіту по роботі

Звіт по роботі повинен містити наступні матеріали:

1. Вихідні дані для розробки програми (завдання).
2. Роздруківки всіх графів станів (первинників) і таблиці символів.
3. Описи створених графів станів.

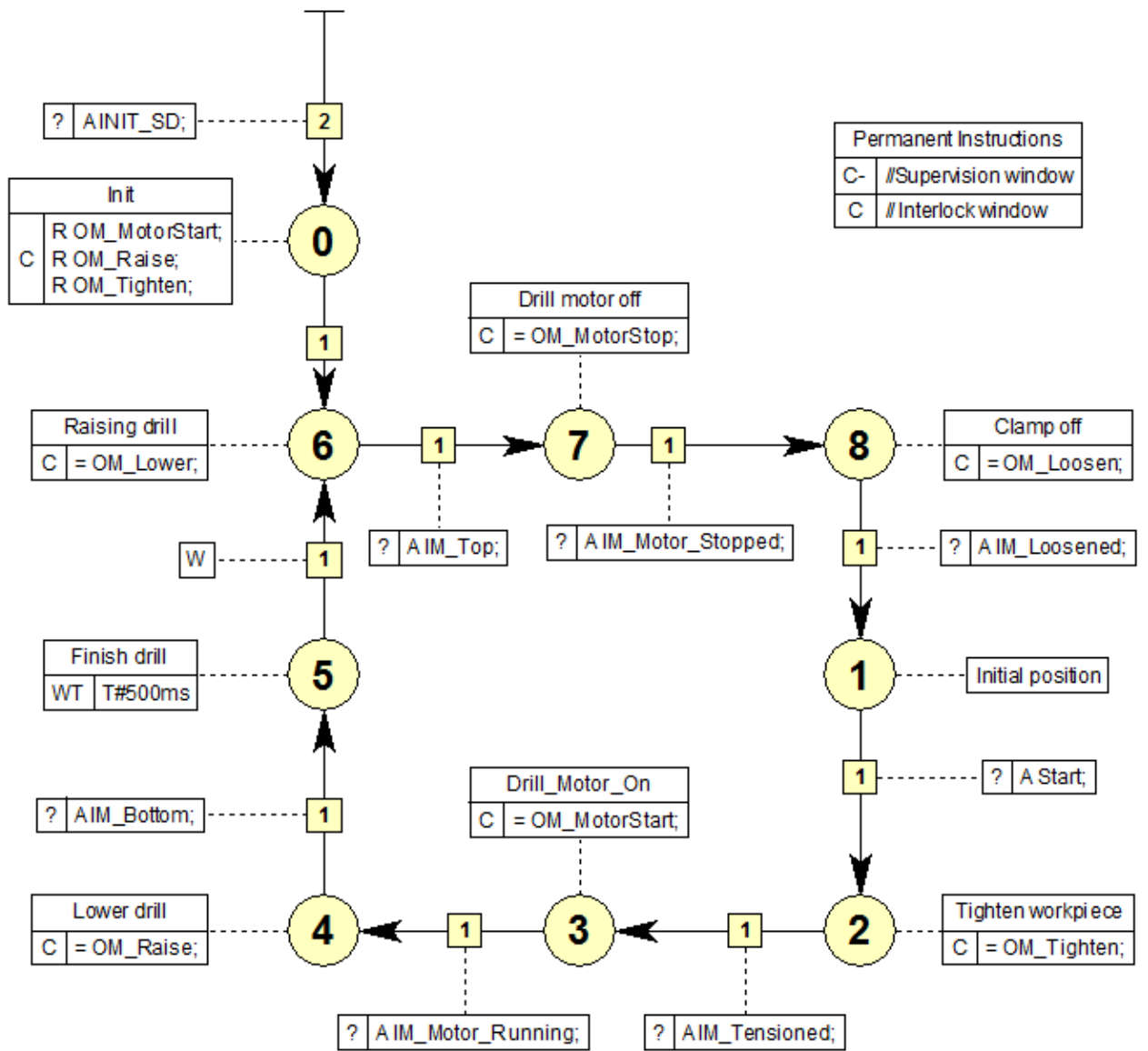


Рисунок 5.12 – Вид координуючого графа станів для свердлильного верстата

6 РОЗРОБКА Й НАЛАГОДЖЕННЯ ПРОГРАМИ HiGRAPH

Ціль роботи: освоєння приймань і придбання навичок створення програм керування встаткуванням у редакторі S7-HiGraph – інструментальному додатку програмної системи STEP 7.

6.1 Створення групового графа

Груповий граф визначає задану послідовність запитів до графів станів, які виконуються циклічно. Запит до графа стану відомий як запит до первинника. Первинники обробляються в програмувальному контролері як змінні групового графа області STAT.

Для створення групового графа потрібно виконати наступне.

В Simatic Manager відкрити папку первинників Source Files із графами станів. Потім клацнути на порожньому місці правою кнопкою й у контекстному меню вибрати команду Insert New Object ► Graph group. Вставленому груповому графові слід привласнити ім'я, наприклад, Drilling_machine. Присвоєння імені здійснюється у вікні Object Properties, яке відкривається за допомогою контекстного меню.

Створити груповий граф можна у вікні S7-HiGraph. Для цього потрібно відкрити будь-який граф станів і вибрати в меню File ► New Graph Group.

Вставка первинників (графів станів).

Створений груповий граф відкривається з порожньою робочою областю. Для вставки в цю область графів станів потрібно клацнути правою кнопкою на порожньому полі й у контекстному меню вибрати Insert Instance. При цьому відкриється вікно вибору файлу “Open”, у якому відображаються всі створені до цього графи станів. Першим вибирається координуючий граф. Закривши вікно “Open” кнопкою ОК, потрібно вставити цей граф на робоче поле групового графа. Вставлений граф станів відображається у вікні групового графа прямокутником з іменем і номером. Імена вставлених графів станів відображаються, як змінні, в області оголошення STAT.

Операції вставки потрібно повторити для всіх створених графів станів. На рисунку 6.1 показане вікно групового графа із вставленими графами станів, створеними для розглянутого тут прикладу.

Установка послідовності виконання процесу.

Послідовність запуску графів станів задається у вікні Run Sequence, яке відкривається однойменною командою контекстного меню. Визначити цю послідовність можна по координуючому графу станів. Для розглянутого прикладу прийнята послідовність виконання: Drilling (1), Motor (2), Feed (3), Vice (4).

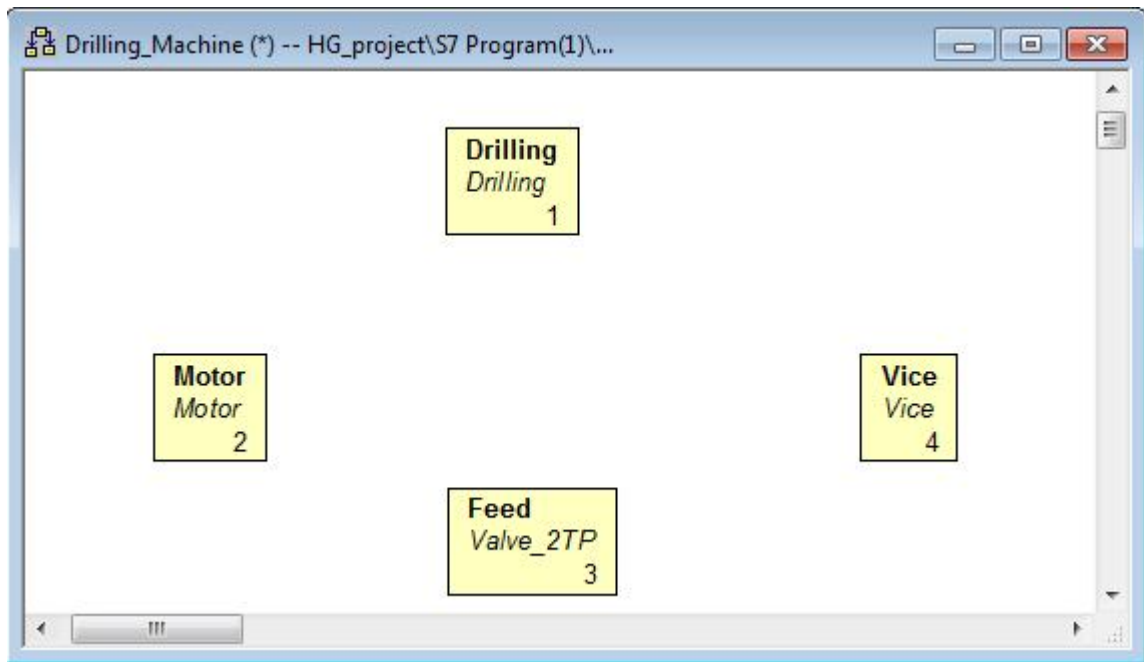


Рисунок 6.1 – Вид вікна групового графа «Drilling_Machine» із вставленими графами станів

Призначення поточних параметрів.

Графи станів для окремих функціональних одиниць являють собою первинники, які можна вставляти в будь-які проекти програм. Зазвичай розроблювач графа дає змінним ті імена, які відбивають сутність елементарного процесу керування, наприклад, для змінної «Включити мотор» призначає ім'я “Motor_On”. Однак, при створенні програми керування конкретним устаткуванням, у якому є кілька моторів, буде потрібна інша система іменування змінних. Тому в груповому графі передбачене зв'язування вихідних імен змінних вставлених графів станів зі змінними, які призначені в створюваній програмі. Процедура зв'язування називається призначенням поточних параметрів. Вона зводиться до наступного.

У вікні групового графа перебувають вставлені графи станів (первинники). У меню View вибираємо команду Details, яка відкриває в нижній області екрана вікно редагування. Це вікно містить вкладку "Current parameters". Виділіть один із графів стану й відкрийте цю вкладку з відображенням списку всіх змінних для обраного графа станів. Далі відкрийте таблицю ідентифікаторів командою Options ► Symbol Table і зробіть оголошення всіх вхідних і вихідних змінних із вказівкою їх фактичних адрес і типів даних. Приклад заповнення таблиці представлений на рисунку 6.2.

Для того, щоб призначити нове ім'я змінній (це ім'я визначене в таблиці символів), потрібно на вкладці "Current parameters" вибрати ім'я в стовпці Name, а в стовпці Current parameters правою кнопкою викликати команду контекстного меню Insert Symbol/Message.

При цьому відкриється список змінних таблиці Symbols, у якому потрібно вибрати відповідну змінну й натиснути Enter (див. рисунок 6.3).

Status	Symbol /	Address	Data type	Comment
1	Clamp_Workpiece	Q 0.3	BOOL	Clamp/hold workpiece with set tension
2	CYCL_EXC	OB 1	OB 1	
3	DB_GG_Drilling	DB 1	DB 1	DB for drilling graph group
4	Drill_at_Bottom	I 0.2	BOOL	Limit switch for "drill at lowest position"
5	Drill_at_Top	I 0.3	BOOL	Limit switch for "drill at highest position"
6	Drill_Motor_On	Q 0.0	BOOL	Switch on drill motor
7	Drill_Motor_Run...	I 0.0	BOOL	Feedback signal for "drill running at set speed"
8	Drill_Motor_Stop...	I 0.1	BOOL	Feedback signal for "drill stationary"
9	GG_Drilling	FC 1	FC 1	FC for drilling graph group
1	Lower_Drill	Q 0.1	BOOL	Lower drill via feed to lowest limit
1	Raise_Drill	Q 0.2	BOOL	Raise drill via feed to highest limit
1	Start_Button	I 0.7	BOOL	Start button for drilling machine
1	Tension_Reached	I 0.4	BOOL	Feedback signal for "workpiece set tension reached"
1				

Рисунок 6.2 – Приклад заповнення таблиці Symbols

The diagram shows three variable symbols: Motor_1 (Motor, 2), Vice_1 (Vice, 4), and Valve2TP_1 (Valve_2TP, 3). Below is a screenshot of the software interface showing a table of variable declarations for 'Drilling_Machine -- HG_project\S7 Program(1)\... [Motor_1]':

Name	Data Type	Current parameter	Message	Comment
Motor_Running	BOOL	"Drill_Motor_Running"		
Motor_Stopped	BOOL	"Drill_Motor_Stopped"		
Motor_On	BOOL	"Drill_Motor_On"		
IM_MotorStart	BOOL			
IM_MotorStop	BOOL			
OM_Motor_Running	BOOL			
OM_Motor_Stopped	BOOL			

A context menu is open over the table, showing the following items:

- Drill Motor On
- Drill_Motor_On (BOOL)
- Drill_Motor_Running (BOOL)
- Drill_Motor_Stopped (BOOL)

Рисунок 6.3 – Відображення процедури заміни імені змінної

Призначення поточних параметрів необхідно виконати для всіх змінних у всіх графах станів, причому в стовпець "Current parameters" потрібно внести навіть ті імена, які не мають відмінності від стовпця "Name". Нові поточні параметри не призначаються тільки для змінних типу "in".

Програмування повідомлень.

Повідомлення – це бінарна змінна, яка встановлюється графом станів передавачем і обробляються в транзакціях графа станів приймача. У транзакціях приймача програмується також дія, яка скидає біт отриманого повідомлення. Повідомлення служать засобами зв'язку між графами станів і використовуються для координації взаємодії графів стану в груповому графі.

Залежно від області дії використовуються два типи повідомлень:

Internal message – внутрішнє повідомлення для зв'язку між графом станів і груповим графом. Зв'язок здійснюється через бітову адресу блоку даних DB групового графа. Цей вид повідомлень використовується при створенні програми керування з одним груповим графом.

External message – зовнішнє повідомлення для зв'язку між графами станів, що перебувають у різних групових графах, або між HiGraph-функціями FC і іншими програмами. Зв'язок здійснюється через загальнодоступну бітову адресу, установлену програмістом.

У якості повідомлень використовуються булеві змінні, оголошені як Message Type. При оголошенні змінних разом з іменем слід указати ознаки – вихідне повідомлення позначити OM (output message), вхідне повідомлення позначити IM (input message).

Для пояснення механізму обміну повідомленнями розглянемо приклад. Нехай зі стану 3 (рис. 5.5) графа-передавача “Feed” у координуючий граф Drilling передається *вихідне* повідомлення OM_Bottom (досягнуте дно). Поточний параметр цього повідомлення записується у вікні Current parameter групового графа з адресою й іншим типом, тобто так: Drilling.IM_Bottom.

Координуючий граф Drilling ухвалює це *вхідне* повідомлення й використовує його в транзакції перемикавання зі стану 4 у стан 5 (рис. 5.12), де передбачена логічна операція I між цим повідомленням і умовою переходу, що перебуває в акумуляторі (команда: A IM_Bottom). Якщо RLO цієї операції буде рівним «1», відбудеться перехід у стан 5, з якого після закінчення часу очікування W буде виконане перемикавання в стан 6. Тут процес піде в іншому напрямку – уже координуючий граф Drilling повинен передати вихідне повідомлення OM_Lower, поточним параметром якого буде: Feed.IM_Lower. Граф Feed ухвалює це вхідне повідомлення й використовує його в транзакції перемикавання в стан 4 (команда A IM_Lower).

Таким чином, вихідні повідомлення відправляються зі станів, а вхідні використовуються в транзакціях.

Вхідні (in) і вихідні (out) повідомлення призначаються у вікні оголошення змінних Variables групового графа в стовпці Message Type. На рисунку 6.4 показані вхідні й вихідні повідомлення для графа станів Motor.

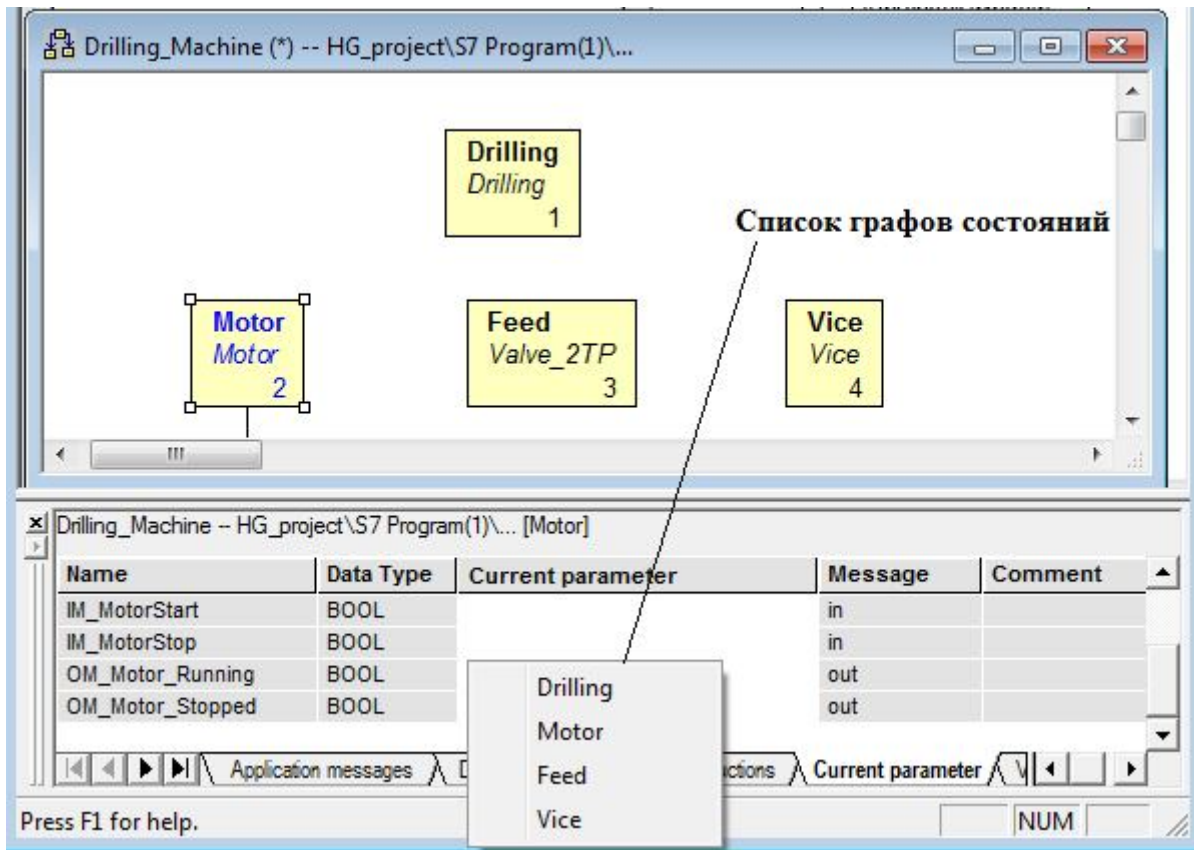


Рисунок 6.4 – Вид вікна оголошення змінних із вхідними й вихідними повідомленнями

Слід урахувати, що для змінних типу OUT програмуються дії присвоєння, наприклад, = OM_Motorstart (рис. 5.12, стан 3) або установки, наприклад, S OM_Motorstart, а для вхідних змінних типу IN програмуються умови, наприклад, A IM_Bottom (перехід у стан 5).

Призначення поточних параметрів для вихідних повідомлень здійснюється у вікні Current parameter групового графа. Процес призначення полягає в наступному.

Спочатку необхідно виділити граф станів. Далі у вікні оголошення змінних прокрутіть список змінних і знайдіть вихідні повідомлення з типом «out». На перетинанні імені повідомлення й стовпця «Current parameter» клацанням правої кнопки викличе контекстне меню, у якому виберіть команду Insert Symbol/Message. По цій команді відкривається невелике вікно зі списком графів, вставлених у груповий граф. На наведеному вище рисунку 6.4 показаний список графів для передачі вихідного повідомлення OM_Motor_Running.

Вважаючи на те, що повідомлення OM_Motor_Running повинне відправлятися в координуючий граф станів Drilling, клацаємо по імені Drilling і це ім'я вставляється редактором у гніздо таблиці. Далі додаємо до імені крапку й редактор автоматично виводить список змінних для призначення цьому вихідному повідомленню поточного параметра (рис. 6.5).

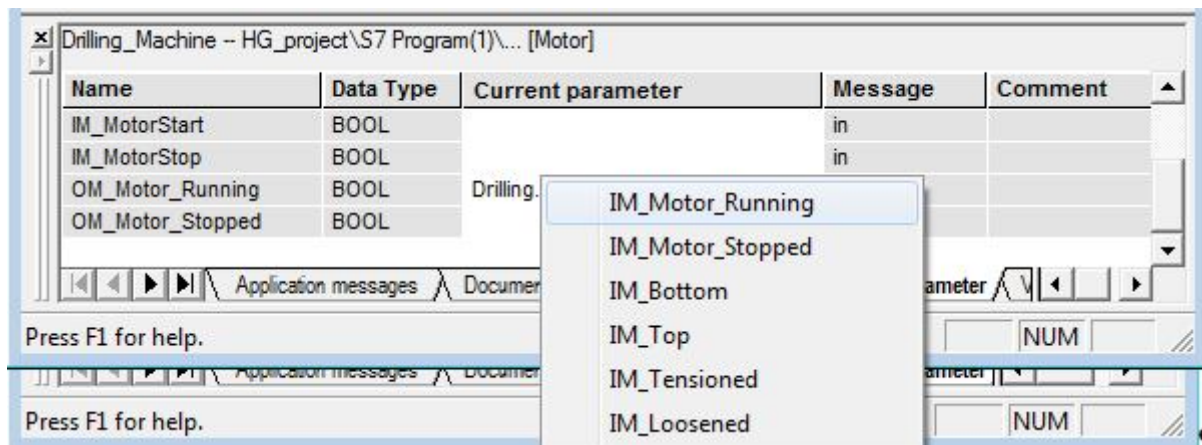


Рисунок 6.5 – Вид вікна для призначення поточного параметра

Вибираємо зі списку IM_Motor_Running і одержуємо результат – Drilling.IM_Motor_Running. Після призначення всіх повідомлень груповий граф приймає вид, показаний на рисунку 6.6.

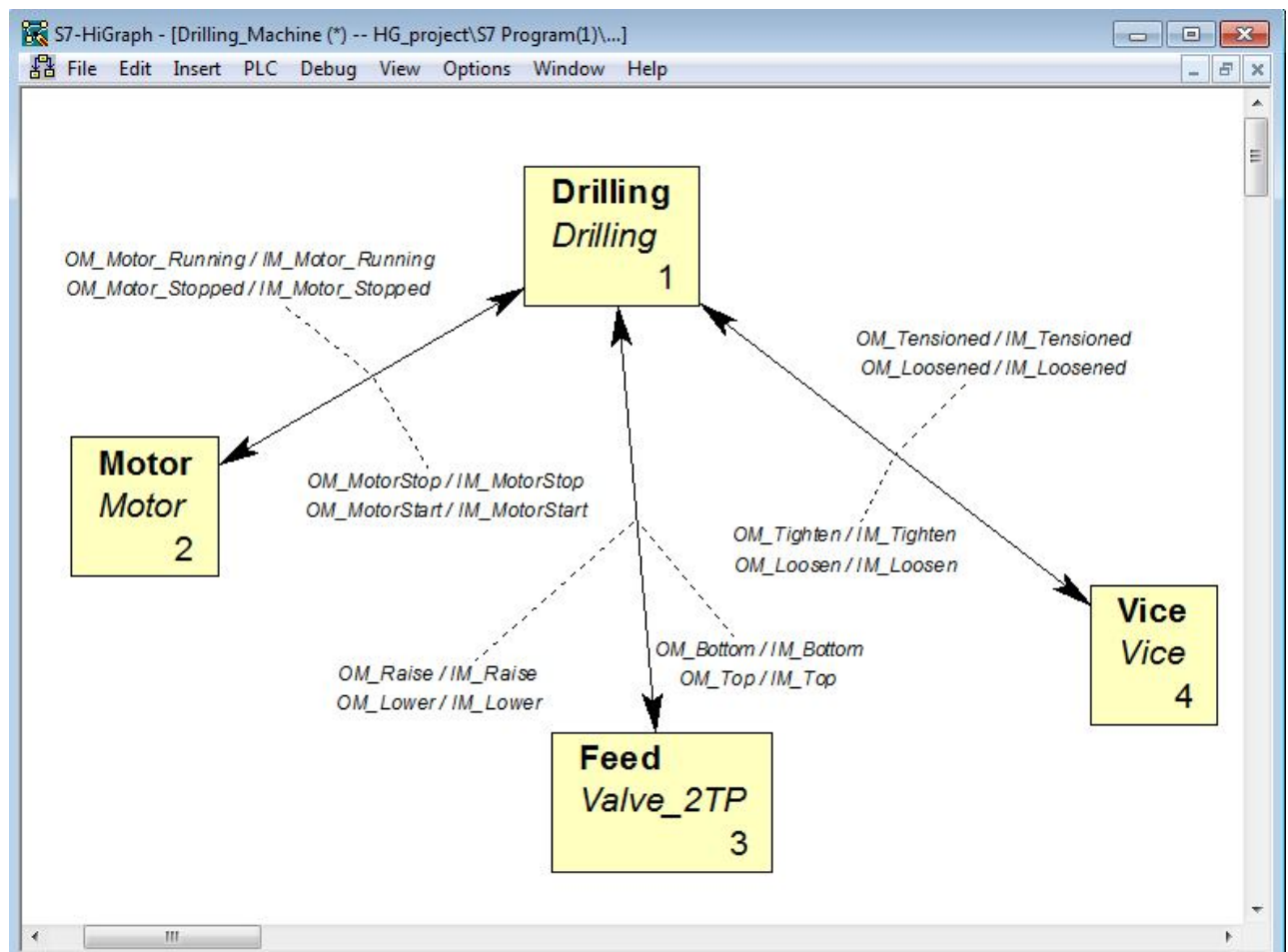


Рисунок 6.6 – Вид групового графа після призначення всіх повідомлень

6.2 Компіляція первинників і створення блоків програми

При збереженні графів станів у контейнері "Source Files" програми S7 ніякої перевірки синтаксису не проводиться. Тому процес редагування може проводитися протягом будь-якого числа сеансів роботи. По закінченню роботи потрібно зберегти об'єкти командою File ⇒ Save.

Процес компілювання застосовується тільки для групових графів, а індивідуальні графи станів не компілюються. При компіляції груповий граф спочатку зберігається, потім HiGraph перевіряє синтаксис програми, створює функцію (FC) і блок даних (DB), а потім запам'ятовує їх у контейнері "Blocks" програми S7.

Якщо при компіляції виявлені синтаксичні помилки, то вони будуть відображені у вікні повідомлення блоку і він не буде створений. Якщо після перевірки синтаксису з'являться тільки попередження, то логічний блок буде створений.

Для компіляції групового графа необхідно виконати наступне.

У меню Options вибрати команду «*ім'я_групового_графа* Settings». У вікні, що відкрилося, перейти на закладку Compile (рис. 6.7) і ввести ім'я функції (FC) і блоку даних (DB). При призначенні імені можна вказувати абсолютне ім'я, наприклад, FC 1. Тут же треба встановити інші параметри налаштування.

Опції налаштувань забезпечують наступне:

- Restructure data block – блок даних DB створюється в процесі трансляції.
- Generate reference data – довідкові дані генеруються автоматично.
- Memory reserves (words) in data block – встановлюється резерв для додаткових графів станів і повідомлень.
- Switch Any transitions once only – опція запобігає повторному перемиканню транзакції запуску.
- Execute cyclic actions with RLO=0 – опція змушує виконати циклічні дії до виходу зі стану при RLO=0.
- Include preceding cyclic actions in entry cycle – опція змушує виконати попередні до входу в стан дії (C-).

Після налаштувань на вкладці Compile слід перемкнутися на вкладку Diagnostics, встановити прапорець на опцію «Format converter diagnostics» і закрити вікно.

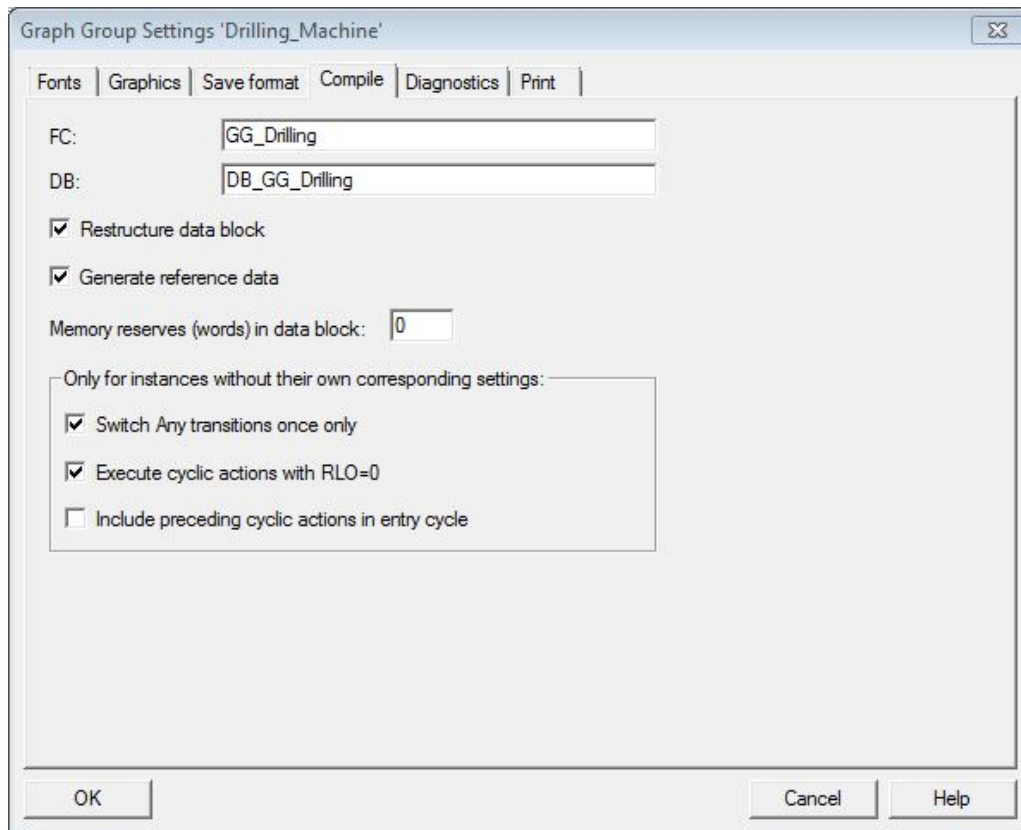


Рисунок 6.7 – Вид вікна настроювання на закладці *Compile*

Для переходу до процесу компіляції потрібно вибрати команду меню File ⇒ Compile. Процес компіляції відображається у вікні повідомлень (нижня область екрана). При виводі помилок потрібну позицію можна знайти, двічі клацнувши на повідомленні про помилки. Після виправлення помилок слід перекомпілювати груповий граф.

На рисунку 6.8 показаний вид вікна групового графа з результатами компіляції.

Для циклічної обробки програми HiGraph у програмувальному контролері вона повинна викликається з організаційного блоку OB1.

Блок OB1 можна програмувати в редакторі базового пакета STEP 7 LAD/STL/FBD. Функція (FC), створена в HiGraph, має вхідний параметр "INIT_SD". Цей параметр повинен установлюватися в "1" при включенні контролера, інакше графи станів у груповому графові ініціалізуватися не будуть. Сигнал установки "INIT_SD" може бути сформований з використанням стартової інформації OB1 (змінна #OB1_SCAN_1) і збережений у тимчасовій змінній OB1, наприклад, в #startup.

Коли блок OB1 вставляється в програму, необхідні змінні вже оголошені в розділі TEMP і залишається дописати цей розділ стартовою змінною #startup, як показано на рисунку 6.9.

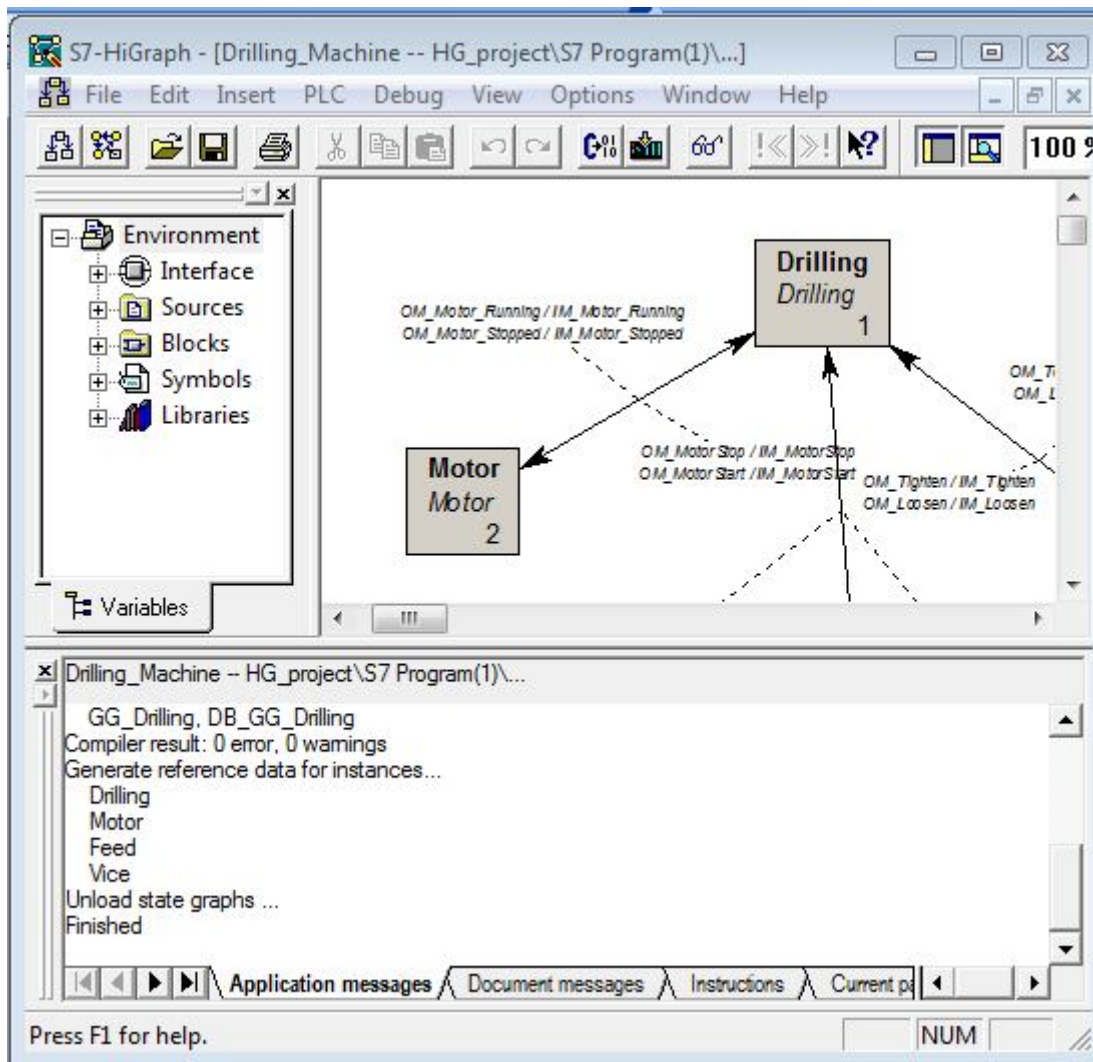


Рисунок 6.8 – Вид вікна групового графа з результатами компіляції

Contents Of: 'Environment\Interface\TEL		Name	Data Type	Address
Interface	TEMP	OB1_EV_CLASS	Byte	0.0
		OB1_SCAN_1	Byte	1.0
		OB1_PRIORITY	Byte	2.0
		OB1_OB_NUMBR	Byte	3.0
		OB1_RESERVED_1	Byte	4.0
		OB1_RESERVED_2	Byte	5.0
		OB1_PREV_CYCLE	Int	6.0
		OB1_MIN_CYCLE	Int	8.0
		OB1_MAX_CYCLE	Int	10.0
		OB1_DATE_TIME	Date...	12.0
		Startup	Bool	20.0

Рисунок 6.9 – Розділ оголошення змінних блоку OB1

Програма організаційного блоку OB1 містить у собі генерацію біта запуску, виклик функції FC1 і ініціалізацію змінної INIT_SD:


```

L OB1_SCAN_1      // Генерація біта запуску.
L 1
==I
= #startup        //Установка першого циклу OB1.
CALL "GG_Drilling" //Виклик функції FC1.
INIT_SD:=#startup //Ініціалізація сигналу установки.

```

6.3 Завантаження програми в контролер та її налагодження

Для завантаження програми користувача в контролер, повинні бути виконані наступні вимоги:

- Програма, яка буде завантажуватися, повинна бути відкомпільована без помилок.
- Повинен бути запрограмований виклик HiGraph FC із циклічно виконуваного блоку.
- Пристрій програмування й програмувальний контролер повинні бути з'єднані.

Завантаження програми здійснюється в наступній послідовності:

1. Установіть CPU у режим STOP.
2. Відкрийте HiGraph програму в SIMATIC Manager.
3. Виберіть необхідні блоки в контейнері блоків:
 - HiGraph FC;
 - HiGraph DB;
 - блок виклику (OB або FB);
 - HiGraphErrEmitterFB (FB20), якщо потрібна діагностика;
 - HiGraphMsgEmitterFC (FC101), якщо потрібна діагностика.
4. Виберіть команду меню PLC ⇒ Download.

Для завантаження в програмувальний контролер тільки функції (FC) з пов'язаним з нею блоком даних (DB) необхідно при відкритому груповому графі вибрати команду меню PLC ⇒ Download і в діалоговому вікні "Download" вибрати завантаження блоку даних DB разом з функцією FC.

Існує можливість контролю й зміни програми, у той час як вона виконується в CPU. Це дозволяє знайти помилки, які не були відображені формальною логікою перевірки при створенні програми або при перевірці синтаксису протягом трансляції.

Редактор HiGraph дозволяє виявити наступні помилки:

- Програмні помилки, наприклад, неправильно встановлений контрольний час.
- Логічні помилки в структурі програми, які означають що,

запрограмовані стани й транзакції не відповідають необхідній послідовності технологічних операцій.

Слід урахувати, що функція налагодження сповільнює проходження програми й може привести до збоїв або перевищення часу циклу.

Доступні наступні функції налагодження й контролю:

- Контроль стану програми.
- Контроль і зміна значень змінних.
- Оцінка довідкових даних.

Перш, ніж використовувати контролюючі функції, повинні бути виконані наступні вимоги:

- Пристрій програмування повинен бути інтерактивно пов'язаний з CPU.
- Програма, повинна бути відкомпільована без помилок.
- Програма (включаючи FC, DB, і OB1) повинна бути завантажена в CPU.
- CPU повинен бути в режимі RUN (читання) або в режимі RUN-P (читання й запис).
- HiGraph-функція FC повинна викликатися із блоку OB1.

Просування програми через індивідуальні стани й транзакції, а також поточна інформація щодо оброблюваних команд показується на екрані. Вікна HiGraph надають наступні можливості контролю:

- У вікні групового графа видні стани й усі первинники групового графа, причому поточний стан відображений у кожному графі.
- У вікні графа станів активний стан позначений кольором, а транзакція, яка привела до цього стану, а також попередній активний стан затінені.

Щоб запустити контроль стану програми, необхідно:

1. При відкритому груповому графі вибрати команду меню Debug ► Monitor для відображення стану групового графа.
2. Вибрати один або декілька первинників, а потім команду меню Edit ► Open Object. Кожний обраний первинник буде відкритий інтерактивно. При цьому відображається детальна інформація стану.
3. Таблиця з інформацією стану відображається спочатку для переміщення з найвищим пріоритетом, який іде від активного стану. Якщо потрібно, можна вибрати іншу активну таблицю команд, щоб відобразити її інформацію.
4. Щоб вибрати для контролю інші первинники, потрібно повернутися до короткого огляду стану групового графа, вибрати первинник й знову застосувати команду меню Edit ► Open Object.

5. Щоб вийти з відображення стану програми, слід дезактивувати команду меню Debug ► Monitor.

Для редагування змінних потрібно вибрати команду меню Debug ► Select Variable. У діалоговому вікні, що відкрилося, треба вибрати необхідні первинники і їх змінні, а після редагування натиснути кнопку "ОК".

При налагодженні програми можна використовувати різні довідкові дані. Для їхнього перегляду слід вибрати команду Options ► Reference.

6.4 Порядок виконання роботи й вимоги до звіту

При виконанні роботи слід створити груповий граф, для якого призначити поточні параметри й повідомлення. Після цього зробити компіляцію створеного групового графа і запрограмувати організаційний блок OB1.

Результати цієї роботи представляються у звіті, який повинен містити роздруківки групового графа, скомпільовану таблицю символів, програму блоку OB1, а також структури блоків S7-програми.

Завантаження програми в контролер для перевірки її працездатності й налагодження може бути виконано тільки в тому випадку, коли первинники ретельно пророблені (оцінка за першу роботу більш 90 балів по 100-бальній шкалі).

Додаток А

Варіанти індивідуальних завдань для роботи 1

Таблиця А.1

Варіант	Базова стійка S7-400								
	Кількість входів				Кількість виходів				Комунікації
	Дискретні		Аналог.		Дискретні		Аналогові		
	=24В	~110В	U	I	=24В	~220В	±10В	±20ma	
1	80	40	25	5	60	25	5	12	Ptp, DP
2	50	25	10	8	80	20	10	4	DP
3	40	60	15	10	70	15	2	6	Ethernet, DP
4	100	35	20	12	50	35	4	5	MPI, DP
5	45	50	25	15	30	30	8	10	DP
6	70	20	10	20	90	10	12	2	Ethernet, DP
7	60	40	15	25	60	25	18	4	Ptp, DP
8	50	25	20	5	80	20	5	8	DP
9	80	60	25	8	70	15	10	12	Ethernet, DP
10	50	35	10	10	50	35	2	18	MPI, DP
11	40	50	15	12	30	30	4	12	DP
12	100	20	20	15	90	10	8	4	Ethernet, DP
13	45	40	25	20	60	25	12	6	MPI, DP
14	70	25	10	25	80	20	18	5	DP
15	60	60	15		70	15		10	Ethernet, DP
16	50	35	20		50	35		2	Ptp, DP
17	90	50	25	5	30	30	5	4	DP
18	80	20	10	8	90	10	10	8	Ethernet, DP
19	50	40	15	10	60	25	2	12	MPI, DP
20	40	25	20	12	80	20	4	18	DP
21	100	60	25	15	70	15	8	12	Ethernet, DP
22	45	35	10	20	50	35	12	4	Ptp, DP
23	70	50	15	25	30	30	18	6	DP
24	60	20	20		90	10		5	Ethernet, DP
25	50	70	30		35	40		10	MPI, DP

Таблиця А.2

Варіант	Відстань до базової стійки, м	Модулі стійки розширення S7-400									
		FM	Дискретні				Аналогові				
			Входів		Виходів		Входів			Виходів	
			+24В	~110В	+24В	~220В	U	I	T°	U	I
1	1	+	80	60	120	150	8	4	15	10	25
2	2		50	35	80	90	12	8	5	25	5
3	3		40	50	70	70	6	15	25	8	12
4	4	+	100	20	65	75	10	20	35	45	20
5	5	+	45	40	50	80	30	7	5		
6	8		90	30	120	150				20	15
7	12		70	15	80	90	8	4	15	10	25
8	20	+	80	60	70	70	12	8	5	25	5
9	1	+	50	35	65	75	6	15	25	8	12
10	2		40	50	50	80	10	20	35	45	20
11	3		100	20	120	150	30	7	5		
12	4	+	45	40	80	90				20	15
13	5	+	90	30	70	70	8	4	15	10	25
14	8		70	15	65	75	12	8	5	25	5
15	12		80	60	50	80	6	15	25	8	12
16	20	+	50	35	120	150	10	20	35	45	20
17	1	+	40	50	80	90	30	7	5		
18	2		100	20	70	70				20	15
19	3		45	40	65	75	8	4	15	10	25
20	4	+	90	30	50	80	12	8	5	25	5
21	5	+	70	15	120	150	6	15	25	8	12
22	8		50	40	80	90	10	20	35	45	20
23	12		20	100	70	70	30	7	5		
24	20	+	35	60	65	75				20	15
25	50		25	40	50	80	18	12	8	12	18

Додаток Б
Варіанти індивідуальних завдань для роботи 2

Таблиця Б.1

№ Вар.	Master (S7-400)				Slave (S7-300)		
	CPU	SM		Інтерфейс	CPU	SM	
		421	422			321	322
1	413-1	16xAC120V 32xDC24V	25xDC24V	DP, Ptp	315-2DP	30xDC24V	20xDC24V
2	413-2DP	30xAC120V 40xDC24V	35xDC24V	DP, Ethernet	313C-2DP	40xDC24V	35xDC24V
3	412-1	10xAC120V 32xDC24V	45xDC24V	DP	314C-2DP	50xDC24V	45xDC24V
4	412-2DP	24xAC120V 50xDC24V	55xDC24V	DP, Ptp	316-2DP	60xDC24V	15xDC24V
5	414-1	16xAC120V 32xDC24V	15xDC24V	DP, Ethernet	317-2	70xDC24V	20xDC24V
6	414-2DP	30xAC120V 40xDC24V	20xDC24V	DP	318-2	30xDC24V	20xDC24V
7	414-3DP	10xAC120V 32xDC24V	30xDC24V	DP, Ptp	315-2DP	40xDC24V	35xDC24V
8	416-1	24xAC120V 50xDC24V	40xDC24V	DP, Ethernet	313C-2DP	50xDC24V	45xDC24V
9	416-2DP	16xAC120V 32xDC24V	25xDC24V	DP	314C-2DP	60xDC24V	15xDC24V
10	416-3DP	30xAC120V 40xDC24V	35xDC24V	DP, Ptp	316-2DP	70xDC24V	20xDC24V
11	413-1	10xAC120V 32xDC24V	45xDC24V	DP, Ethernet	317-2	30xDC24V	30xDC24V
12	413-2DP	24xAC120V 50xDC24V	55xDC24V	DP	318-2	40xDC24V	35xDC24V
13	412-1	16xAC120V 32xDC24V	15xDC24V	DP, Ptp	315-2DP	50xDC24V	45xDC24V
14	412-2DP	30xAC120V 40xDC24V	20xDC24V	DP, Ethernet	313C-2DP	60xDC24V	15xDC24V
15	414-1	10xAC120V 32xDC24V	30xDC24V	DP	314C-2DP	70xDC24V	20xDC24V
16	414-2DP	24xAC120V 50xDC24V	40xDC24V	DP, Ptp	316-2DP	30xDC24V	30xDC24V
17	414-3DP	16xAC120V 32xDC24V	25xDC24V	DP, Ethernet	317-2	40xDC24V	35xDC24V
18	416-1	30xAC120V 40xDC24V	15xDC24V	DP	318-2	50xDC24V	45xDC24V
19	416-2DP	10xAC120V 32xDC24V	45xDC24V	DP, Ptp	315-2DP	60xDC24V	15xDC24V
20	416-3DP	24xAC120V 30xDC24V	25xDC24V	DP	313C-2DP	70xDC24V	20xDC24V
21	413-2DP	10xAC120V 40xDC24V	45xDC24V	DP, Ethernet	313C-2DP	24xDC24V	15xDC24V
22	414-2DP	30xAC120V 40xDC24V	20xDC24V	DP	318-2	36xDC24V	32xDC24V

Таблиця Б.2 – Вихідні дані для конфігурування станцій ET200

№ Вар.	ET 200M			ET 200S			
	SM		FM	Кіл. сигналів		Кіл. приводів	
	321	322		Уведення	Виводу	Реверсивних	Нереверсивних
1	16xDC24V	20xDC24V	350-1	6DIx24V	4DOx24V	2	1
2	30xDC24V	35xDC24V	немає	4DIx120V	6DOx24V	1	2
3	30xDC24V	25xDC24V	353	6DIx24V	4DO реле	2	2
4	24xAC120V	15xDC24V	немає	4DIx24V	4DOx24V	2	0
5	35xDC24V	20xDC24V	немає	4DIx120V	6DOx24V	0	2
6	30xDC24V	45xDC24V	354	6DIx24V	2DO реле	2	1
7	10xAC120V	15xDC24V	немає	2DIx24V	4DOx24V	1	2
8	30xDC24V	20xDC24V	немає	4DIx120V	6DOx24V	2	2
9	30xDC24V	30xDC24V	354	6DIx24V	2DO реле	2	0
10	15xAC120V	35xDC24V	немає	2DIx24V	4DOx24V	2	1
11	30xDC24V	45xDC24V	немає	4DIx120V	6DOx24V	1	2
12	24xAC120V	15xDC24V	353	6DIx24V	2DO реле	2	2
13	30xDC24V	20xDC24V	немає	2DIx24V	4DOx24V	2	0
14	30xDC24V	30xDC24V	немає	4DIx120V	6DOx24V	2	1
15	20xAC120V	35xDC24V	350-1	6DIx24V	2DO реле	1	2
16	30xDC24V	45xDC24V	немає	2DIx24V	4DOx24V	2	2
17	30xDC24V	15xDC24V	немає	4DIx120V	6DOx24V	2	0
18	30xDC24V	20xDC24V	353	6DIx24V	2DO реле	0	2
19	30xDC24V	30xDC24V	немає	2DIx24V	4DOx24V	2	1
20	35xAC120V	35xDC24V	немає	4DIx120V	6DOx24V	1	2
21	30xDC24V	15xDC24V	354	6DIx24V	2DO реле	2	2
22	30xDC24V	20xDC24V	немає	8DIx24V	4DOx24V	2	0

Додаток В

Варіанти індивідуальних завдань по програмуванню для робіт 3 і 4

Варіант	Додаткова функція програми
1	Поштовховий режим включення двигуна
2	Контроль часу запуску двигуна
3	Підрахунок деталей для формування партії
4	Контроль часу транспортування деталі
5	Контроль часу очікування завантаження деталі на транспортер
6	Контроль часу очікування зняття деталі із транспортера
7	Підрахунок числа запусків транспортера для техобслуговування
8	Індикація запуску, останову й техобслуговування
9	Керування в ручному й автоматичному режимах
10	Контроль обриву живлення в ланцюзі двигуна
11	Система керування двома конвеєрами
12	Керування вентилем, за допомогою якого протягом заданого часу рідина подається в тару на позиції завантаження
13	Керування вентилем, за допомогою якого рідина подається в тару на позиції завантаження з контролем її кількості
14	Керування пристроєм гідрозатискача деталі при установці деталі в супутник і розтиском при знятті деталі
15	Завдання швидкості обертання двигуна транспортера
16	Датчики й електродвигун конвеєра підключити до станції ET200S
17	Датчики й електродвигун конвеєра підключити до станції ET200M
18	Увести сигналізацію світлофором із червоним заборонним і зеленим розв'язним світлом. Червоне світло включається, коли на транспортері перебуває деталь, зелений – коли деталі немає.
19	Вивести інформацію про вагу деталі на пульт оператора з перетворенням аналогового сигналу в VCD-формат.
20	Систему керування двома конвеєрами побудувати з використанням двох станцій ET200S

Додаток Г

Базові функції STL

Двійкові логічні операції

A	-	операція И (AND) для перевірки присутності рівня "1"
AN	-	операція И (AND) для перевірки присутності рівня "0"
O	-	операція АБО (OR) для перевірки присутності рівня "1"
ON	-	операція АБО (OR) для перевірки присутності рівня "0"
X		операція Виключаюче АБО (Exclusive OR) для перевірки присутності рівня "1"
XN		операція Виключаюче АБО (Exclusive OR) для перевірки присутності рівня "0"
-	I	вхід
-	Q	вихід
-	M	меркер
-	L	біт в області локальних даних
-	T	функція таймера
-	C	функція лічильника
-	DBX	біт в області глобальних даних
-	DIX	біт в екземплярному DB
-	==0	значення результату операції дорівнює нулю
-	<>0	значення результату операції не дорівнює нулю
-	>0	значення результату операції більше нуля
-	>=0	значення результату операції більше нуля або дорівнює нулю
-	<0	значення результату операції менше нуля
-	<=0	значення результату операції менше нуля або дорівнює нулю
-	UO	значення результату операції невірно
-	OV	переповнення
-	OS	збережене переповнення
-	BR	двійковий результат
A(операція И (AND) з відкриваючою дужкою
AN(операція И (AND) з відкриваючою дужкою
O(операція АБО (OR) з відкриваючою дужкою
ON(операція АБО (OR) з відкриваючою дужкою
X(операція, Що Виключає АБО (Exclusive OR) з відкриваючою дужкою
XN(операція, Що Виключає АБО (Exclusive OR) з відкриваючою дужкою
)		закриваюча дужка
O		операція АБО (OR), що поєднує операції И (AND)
NOT		операція заперечення RLO
SET		операція установки RLO
CLR		операція скидання RLO
SAVE		операція фіксації RLO в BR

Операції з пам'яттю

=	-	операція присвоєння
S	-	операція установки
R	-	операція скидання
FP	-	позитивний фронт сигналу
FN	-	негативний фронт сигналу
-	I	вхід
-	Q	вихід
-	M	меркер
-	L	біт в області локальних даних
-	DBX	біт в області глобальних даних
-	DIX	біт в екземплярному DB

Функції передачі

L	-операція завантаження (load)
T	-операція передачі (transfer)
-	IB вхідний байт
-	IW вхідне слово
-	ID вхідне подвійне слово
-	QB вихідний байт
-	QW вихідне слово
-	QD вихідне подвійне слово
-	MB байт меркерів
-	MW слово меркерів
-	MD подвійне слово меркерів
-	LB байт локальних даних
-	LW слово локальних даних
-	LD подвійне слово локальних даних
-	DBB байт глобальних даних
-	DBW слово глобальних даних
-	DBD подвійне слово глобальних даних
-	DIB байт в екземплярному DB
-	DIW слово в екземплярному DB
-	DID подвійне слово в екземплярному DB
-	STW слово стану
LPIB	завантаження (load) периферійного вхідного байта
LPIW	завантаження (load) периферійного вхідного слова
LPID	завантаження (load) периферійного вхідного подвійного слова
TPQB	передача (transfer) периферійного вихідного байта
TPQW	передача (transfer) периферійного вихідного слова
TPQD	передача (transfer) периферійного вихідного подвійного слова
L T	"звичайне" завантаження значення таймера
LCT	завантаження значення таймера в BCD-кодi
LC	"звичайне" завантаження значення лічильника
LCC	завантаження значення лічильника в BCD-кодi
L	<i>const</i> завантаження (load) константи
L	P#.. завантаження (load) покажчика
L P#var	завантаження (load) початкової адреси змінної

Функції акумуляторів

PUSH	зрушення вмісту акумуляторів "уперед"
POP	зрушення вмісту акумуляторів "назад"
ENT	зрушення вмісту акумуляторів 2 і 3 "уперед"
LEAVE	зрушення вмісту акумуляторів 3 і 4 "уперед"
TAK	обмін умістом між акумуляторами 1 і 2
CAW	обмін умістом між байтами 0 і 1 акумулятора 1
CAD	обмін умістом між усіма байтами акумулятора 1

Функції таймерів

SPT	запуск таймера в режимі "керованого імпульсу"
SET	запуск таймера в режимі "розширеного імпульсу"
SDT	запуск таймера в режимі "із затримкою включення"
SST	запуск таймера в режимі "із затримкою включення з пам'яттю"
SFT	запуск таймера в режимі "із затримкою вимикання"
R T	скидання таймера
FR T	дозвіл перезапуску таймера

Функції лічильників

CU	C	запуск лічильника в режимі "прямий рахунок"
CD	C	запуск лічильника в режимі "зворотний рахунок"
S	C	установка лічильника
R	C	скидання лічильника
FR	C	дозвіл перезапуску лічильника

Функції для обробки чисел

Функції порівняння

==I	перевірка даних формату INT на рівність
<>I	перевірка даних формату INT на нерівність
>I	порівняння даних формату INT за критерієм "більше чим"
>=I	порівняння даних формату INT за критерієм "більше або рівно"
<I	порівняння даних формату INT за критерієм "менше ніж"
<=I	порівняння даних формату INT за критерієм "менше або рівно"
==D	перевірка даних формату DINT на рівність
<>D	перевірка даних формату DINT на нерівність
>D	порівняння даних формату DINT за критерієм "більше чим "
>=D	порівняння даних формату DINT за критерієм "більше або рівно"
<D	порівняння даних формату DINT за критерієм "менше ніж"
<=D	порівняння даних формату DINT за критерієм "менше або рівно"
==R	перевірка даних формату REAL на рівність
<>R	перевірка даних формату REAL на нерівність
>R	порівняння даних формату REAL за критерієм "більше чим "
>=R	порівняння даних формату REAL за критерієм "більше або рівно"
<R	порівняння даних формату REAL за критерієм "менше ніж"
<=R	порівняння даних формату REAL за критерієм "менше або рівно"

Математичні функції

SIN	синус
COS	косинус
TAN	тангенс
ASIN	арксинус
ACOS	арккосинус
ATAN	арктангенс
SQR	знаходження квадрата числа
SQRT	добування квадратного кореня із числа
EXP	експонента по підставі e
LN	натуральний логарифм

Арифметичні функції

+I	додавання двох чисел формату INT
-I	вирахування двох чисел формату INT
*I	множення двох чисел формату INT
/I	розподіл двох чисел формату INT
+D	додавання двох чисел формату DINT
-D	вирахування двох чисел формату DINT
*D	множення двох чисел формату DINT
/D	розподіл двох чисел формату DINT (ціла частина)
MOD	розподіл двох чисел формату DINT (залишок)
+R	додавання двох чисел формату REAL
-R	вирахування двох чисел формату REAL
*R	множення двох чисел формату REAL
/R	розподіл двох чисел формату REAL
+	<i>const</i> додавання з константою
+	P#.. додавання з покажчиком

DEC *n* декрементування

INC *n* інкрементування

Функції перетворення

ITD конвертування даних формату INT у формат DINT

ITB конвертування даних формату INT у формат BCD

DTB конвертування даних формату DINT у формат DINT

DTR конвертування даних формату DINT у формат REAL

BTI конвертування даних формату BCD у формат INT

BTD конвертування даних формату BCD у формат DINT

Конвертування даних формату REAL у формат DINT, при цьому відбувається:

RND+ округлення даних до найближчого більшого цілого числа

RND- округлення даних до найближчого меншого цілого числа

RND округлення даних до найближчого цілого числа

TRUNC усікання дробової частини числа

INVI знаходження зворотного коду двійкового числа формату INT

INVD знаходження зворотного коду двійкового числа формату DINT

NEGI інвертування числа формату INT

NEGD інвертування числа формату DINT

NEGR інвертування числа формату REAL

ABS знаходження абсолютного значення числа формату REAL

Функції зрушення

SLW - побітове зрушення вліво вмісту молодшого слова акумулятора 1

SLD - побітове зрушення вліво вмісту акумулятора 1

SRW - побітове зрушення вправо вмісту молодшого слова акк. 1

SRD - побітове зрушення вправо вмісту акумулятора 1

SSI - побітове зрушення зі знаком вмісту молодшого слова акк. 1

SSD - побітове зрушення зі знаком вмісту акумулятора 1

RLD - циклічне зрушення вліво вмісту акумулятора 1

RRD - циклічне зрушення вправо вмісту акумулятора 1

- *n* на *n* позицій

- на число позицій, зазначене в акумуляторі Accum 2

RLDA - циклічне зрушення вліво з використанням біта CC1

RRDA - циклічне зрушення вправо з використанням біта CC1

Логічні функції для слів даних

AW - операція И (AND) для слова даних

AD - операція И (AND) для подвійного слова даних

OW - операція АБО (OR) для слова даних

OD - операція АБО (OR) для подвійного слова даних

XOW - операція Виключаюче АБО (Exclusive OR) для слова даних

XOD - операція Виключаюче АБО (Exclusive OR) для подвійного слова даних

- *const* з константою формату слова даних або подвійного слова даних

- із вмістом акумулятора Accum 2

Функції керування в програмі

Функції переходу

JU *мітка* безумовний перехід

Виконується перехід,

JC *мітка* якщо RLO = "1"

JCB *мітка* якщо RLO = "1" зі збереженням RLO

JCN *мітка* якщо RLO = "0"

JNB	<i>мітка</i>	якщо RLO = "0" зі збереженням RLO
JBI	<i>мітка</i>	якщо BR = "1"
JNBI	<i>мітка</i>	якщо BR = "0"
Виконується перехід,		
JZ	<i>мітка</i>	якщо результат = "0"
JN	<i>мітка</i>	якщо результат <> "0"
JP	<i>мітка</i>	якщо результат > "0"
JPZ	<i>мітка</i>	якщо результат >= "0"
JM	<i>мітка</i>	якщо результат < "0"
JMZ	<i>мітка</i>	якщо результат <= "0"
JUO	<i>мітка</i>	якщо результат некоректний
JO	<i>мітка</i>	перехід виконується при переповненні
JOS	<i>мітка</i>	перехід виконується при запомненому переповненні
JL	<i>мітка</i>	розподільник переходів
LOOP	<i>мітка</i>	циклічний перехід

Функції обробки блоків

CALL	FB	виклик функціонального блоку
CALL	FC	виклик функції
CALL	SFB	виклик системного функціонального блоку
CALL	SFC	виклик системної функції
UC	FB	безумовний виклик функціонального блоку
CC	FB	виклик функціонального блоку за умовою
UC	FC	безумовний виклик функції
CC	FC	виклик функції за умовою
BEU		безумовне завершення обробки блоку
BEC		завершення обробки блоку за умовою
BE		безумовне завершення обробки блоку
OPN	DB	виклик глобального блоку даних
OPN	DI	виклик екземплярного блоку даних
CDB		обмін даними між регістрами блоку
L	DBNO	завантаження (load) номера глобального блоку даних
L	DINO	завантаження (load) номера екземплярного блоку даних
L	DBLG	завантаження (load) розміру глобального блоку даних
L	DILG	завантаження (load) розміру екземплярного блоку даних
NOP	0	нуль-операція
NOP	1	нуль-операція
BLD	<i>n</i>	інструкції відображення програми

