В.Ю.Кучерук, В.О.Поджаренко, П.І.Кулаков

ПРОГРАМУВАННЯ ЛОГІЧНИХ КОНТРОЛЕРІВ SCHNEIDER ELECTRIC

Міністерство освіти України
Вінницький державний технічний університет

В.Ю.Кучерук, В.О.Поджаренко, П.І.Кулаков

ПРОГРАМУВАННЯ ЛОГІЧНИХ КОНТРОЛЕРІВ SCHNEIDER ELECTRIC

Навчальний посібник

Вінниця ВДТУ 2002

УДК 621.3:658.562

К 58

Р е ц е н з е н т и:

П.Г.Столярчук, доктор технічних наук, професор
Р.Н.Квєтний, доктор технічних наук, професор
С.В.Павлов, кандидат технічних наук, доцент

Рекомендовано до видання Ученою радою Вінницького державного технічного університету Міністерства освіти і науки України

В.Ю.Кучерук, В.О.Поджаренко, П.І.Кулаков

К 58 **Програмування логічних контролерів Schneider Electric.**
Навчальний посібник. – В.: ВДТУ, 2001. - 134 с.

Навчальний посібник призначений для вивчення теоретичного матеріалу, виконання лабораторних робіт та курсових проектів з дисциплін “Мікроконтролери та ОМЕОМ”, “Основи мікропроцесорної техніки”, “Програмне забезпечення комп’ютерних вимірювальних пристроїв”, “Проектування комп’ютеризованих систем керування” для студентів, що навчаються за спеціальністю 7.091302 “Метрологія та вимірювальна техніка”. Він складається з семи розділів. Коротко розглянуті сучасні системи автоматизованого керування, будова контролера Modicon TSX Micro, структура та алгоритми роботи програмного забезпечення, мови програмування Instruction List, Ladder Diagram, Structured Text, Grafcet.

Перевагою навчального посібника є наявність у ньому конкретних прикладів використання логічних контролерів.

Розрахований для широкого кола інженерно-технічних працівників, а також може бути використаний студентами, які вивчають програмування логічних контролерів.

ЗМІСТ

ВСТУП

Створення фірмою Intel першого мікропроцесора положило початок ері комп′ютеризації. “Завдяки мікропроцесорам комп′ютери стали масовим, загальнодоступним продуктом”, - заявив Тед Гофф (Ted Hoff), один із винахідників першого мікропроцесора. Його ім′я, разом з іменами його коллег – Федеріко Феджина (Federico Faggin) і Стена Мейзора (Stan Mazor), внесено у список лауреатів Національної зали слави винахідників США, а сам винахід визнаний одним із найбільших досягнень XX століття.

В мікропроцесорах – найбільш складних мікроелектронних пристроях – втілені найпередовіші досягнення інженерної думки. В умовах, властивих даній сфері виробництва, жорсткої конкуренції і величезних капіталовкладень випуск кожної нової моделі мікропроцесора пов′язаний з черговим науковим, конструкторським, технологічним проривом.

Використання мікроелектронних засобів в системах автоматизованого керування приводить не тільки до підвищення техніко-економічних показників систем (вартість, надійність, габаритні розміри) і скорочення термінів розробки, а й надають їм принципово нові якості (розширені функціональні можливості, модофікованість, адаптивність тощо).

За останні роки в мікроелектроніці бурхливий розвиток отримав напрямок, пов′язаний з випуском програмовних логічних контролерів (ПЛК). ПЛК представляють собою прилади, конструктивно виконані в одному корпусі і включають у себе всі складові частини мікро-ЕОМ: мікропроцесор, пам′ять програм і пам′ять даних, а також програмовні інтерфейсні схеми для зв′язку із зовнішнім середовищем. Використання ПЛК в системах керування забезпечує досягнення виключно таких високих показників ефективності при низькій вартості.

Для успішної конкуренції, особливо на західних ринках, технологічні процеси повинні оснащуватися найсучаснішими системами керування й електроустаткуванням.

Оснащення сучасних машин, навіть найпростіших, системами керування на базі ПЛК, не просто данина моді, а одне з умов забезпечення нормального їхнього функціонування, збільшення продуктивності і мобільності машин, зниження витрат на їхнє виробництво й обслуговування.

Schneider Electric традиційно орієнтується на створення систем керування і має величезний досвід у розробці подібних систем. Тому підтвердженням служать тисячі реалізацій з використанням ПЛК цього великого міжнародного концерну в металоброблювальному підйомно-транспортному устаткуванні, машинах для легкої і харчової промисловості і навіть бортових систем для залізничного транспорту і важких вантажівок. Яким же повинен бути ПЛК, щоб щонайкраще відповідати сучасним

вимогам? Насамперед - мати високу продуктивність для забезпечення мінімального часу реакції на зовнішні події. Не менш важлива наявність широкої гами модулів введення-виведення, у тому числі і спеціалізованих, - таких як лічильники, модулі керування позиціонуванням, негайної зупинки. Важлива і вартість монтажу і підключення контролера, - адже від цього залежить кінцева вартість виробу. Оскільки машини стають все більш складними, а пристрої автоматики усе більше знижуються в ціні, то в системах керування устаткуванням з'являється усе більше "інтелектуальних" пристроїв - таких як варіатори швидкості і пускачі, засоби людино-машинного інтерфейсу і сигнальні пристрої. І природно, що ПЛК повинен забезпечувати простий і недорогий інтерфейс із усіма такими пристроями. І, нарешті, ПЛК повинен бути дуже надійним, механічно міцним і мати можливості вбудованої діагностики апаратних засобів. Від цього залежить надійність і ремонтопридатність усієї машини чи установки.

Усім перерахованим вище вимогам відповідають контролери гами Modicon TSX, що втілили в собі найпередовіші технологічні досягнення і багаторічний досвід фахівців Schneider Automation.

Вивчення такої наукоємкої предметної області, що так інтенсивно розвивається, як мікроелектроніка, мікропроцесорна і контролерна техніка – задача цікава і складна, яка вимагає постійного вдосконалення і поповнення отримуваних знань і знайомства із суміжними науково-технічними областями.

Навчальний посібник може бути використаний при вивченні дисциплін “Мікроконтролери та ОМЕОМ”, “Основи мікропроцесорної техніки”, “Програмне забезпечення комп’ютерних вимірювальних пристроїв”, “Проектування комп’ютеризованих систем керування” для студентів, що навчаються за спеціальністю 7.091302 “Метрологія та вимірювальна техніка”.

1 ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ СИСТЕМ АВТОМАТИЗОВАНОГО КЕРУВАННЯ

1.1 Основні терміни і поняття автоматизованих систем керування

Розподілена система керування (*Distributed Control System - DCS*) - сукупність універсальних і спеціалізованих промислових обчислювальних засобів - вузлів розподіленої обробки даних, об'єднаних структурованою комунікаційною системою для вирішення задач керування неперервними технологічними процесами в реальному масштабі часу.

Програмовний логічний контролер (*Programmable Logic Controller — PLC*) - програмовний обчислювальний пристрій, спеціально розроблений для вирішення задач збору й обробки аналогових і/чи дискретних сигналів, дикретно-логічного керування технологічним устаткуванням і неперервним регулюванням параметрів технологічних процесів у реальному масштабі часу.

Промисловий персональний комп'ютер (*Industrial PC*) - персональний комп'ютер з IBM PC-подібною чи сумісною архітектурою, адаптований для вирішення задач керування технологічними процесами і промисловим устаткуванням на рівні людино-машинного інтерфейсу (SCADA/HMI чи «верхньому» рівні в дворівневій, вітчизняній класифікації АСУ ТП). Апаратні або програмні удосконалення PC, покликані забезпечити також керування процесом у реальному масштабі часу, дали народження новому напрямку, що одержав назву **PC based control.**

Програмно-технічні засоби, що лежали в основі архітектури DCS і PLC (на відміну від PC based control), спеціально створювалися для виконання функцій промислового контролю і керування в реальному масштабі часу. У даний момент їм належить більше 99% $12-мільярдного світового ринку промислових систем керування.

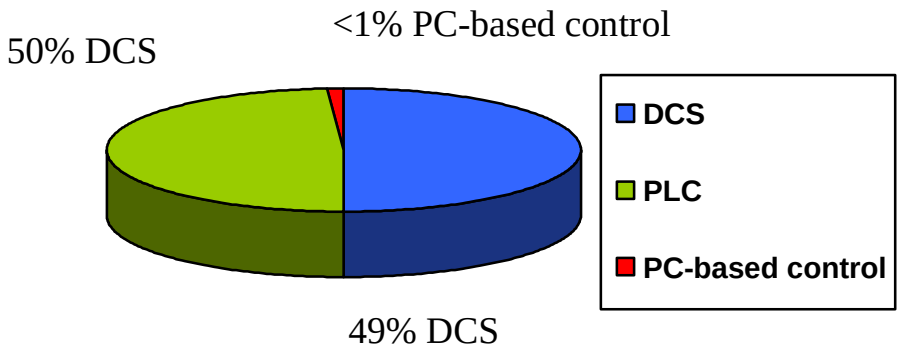


Рисунок 1.1 - Розподіл на світовому ринку засобів автоматизації (журнал JE, квітень 1997р., Франція)

SCADA (*Supervisory Control And Data Acquistion*) – система супервізорного керування і збору даних або система керування і моніторингу, яка вміщує програмно-апаратні засоби, що взаємодіють між собою через глобальні мережі. Система складається з:

- SCADA Host – основного вузла SCADA;
- MTU (*Master Terminal Unit*) – основний термінальний пристрій ("термінальний" – у смислі "кінцевий");
- RTU (*Remote Terminal Unit*) – віддалений термінальний пристрій;
- Комунікаційного обладнання і середовища передачі даних.

SCADA Host – мікропроцесорний пристрій, який використовується для супервізорного контролю-керування і моніторингу в SCADA-системі і організації людино-машинного інтерфейсу. SCADA Host – теж саме, що і SCADA/HMI (*SCADA Human Machine Interface*).

MTU (*Master Terminal Unit*) – мікропроцесорний пристрій, який використовується в SCADA як концентратор даних для SCADA Host, а також може дублювати RTU.

RTU (*Remote Terminal Unit*) – мікропроцесорний пристрій, який традиційно використовується для збору і контролю аналогових і дискретних сигналів і виконання дистанційних команд керування від SCADA Host.

1.2 Історія розвитку засобів автоматизації

Історія DCS і PLC іде своїми коренями в часи кінця 70-х і початку 80-х років, коли поява мікропроцесорів викликала вибухоподібні революційні зміни в комп'ютерній техніці і технологіях, що, у свою чергу, привели до радикального перегляду устояних на той час підходів до промислової автоматизації взагалі й автоматизації неперервних технологічних процесів зокрема. Протягом 80-х автоматизовані системи керування технологічними процесами (АСУ ТП) на таких складних і відповідальних об'єктах з неперервним виробничим циклом, як, наприклад, — атомні електростанції (АЕС), представляли собою царство могутніх (на той час) промислових комп'ютерів — мейнфреймів, що централізовано здійснювали вирішення покладених на них задач інформаційної підтримки керування енергоблоком: збір і обробку сигналів датчиків, внутріреакторний контроль, розрахунок техніко-економічних показників роботи блоку, представлення інформації операторам, накопичення і документування архівних даних. Прикладом такого підходу є блокові інформаційно-обчислювальні системи енергоблоків серії ВВЕР-1000 на більшості АЕС України, де дотепер експлуатуються керуючі обчислювальні комплекси (мейнфрейми) типу СМ-2М. У той самий час, для вирішення задач неперервного регулювання окремих параметрів

процесу і дискретно-логічного керування устаткуванням енергоблоків використовувалася аналогова і релейна автоматика. Радикальні зміни (принаймні за кордоном) відбулися на початку 80-х з появою технологій DCS (рисунок 1.2) і PLC (рисунок 1.3).

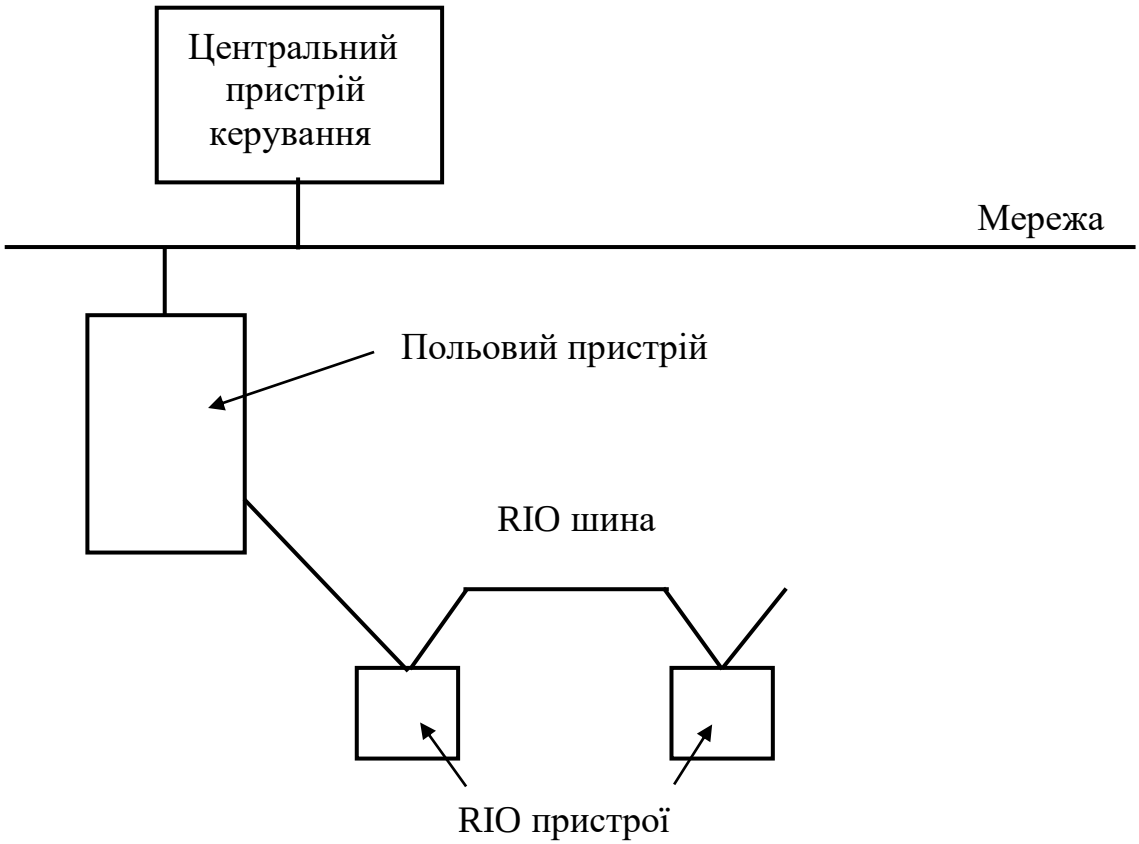


Рисунок 1.2 - Спрощена структура DCS

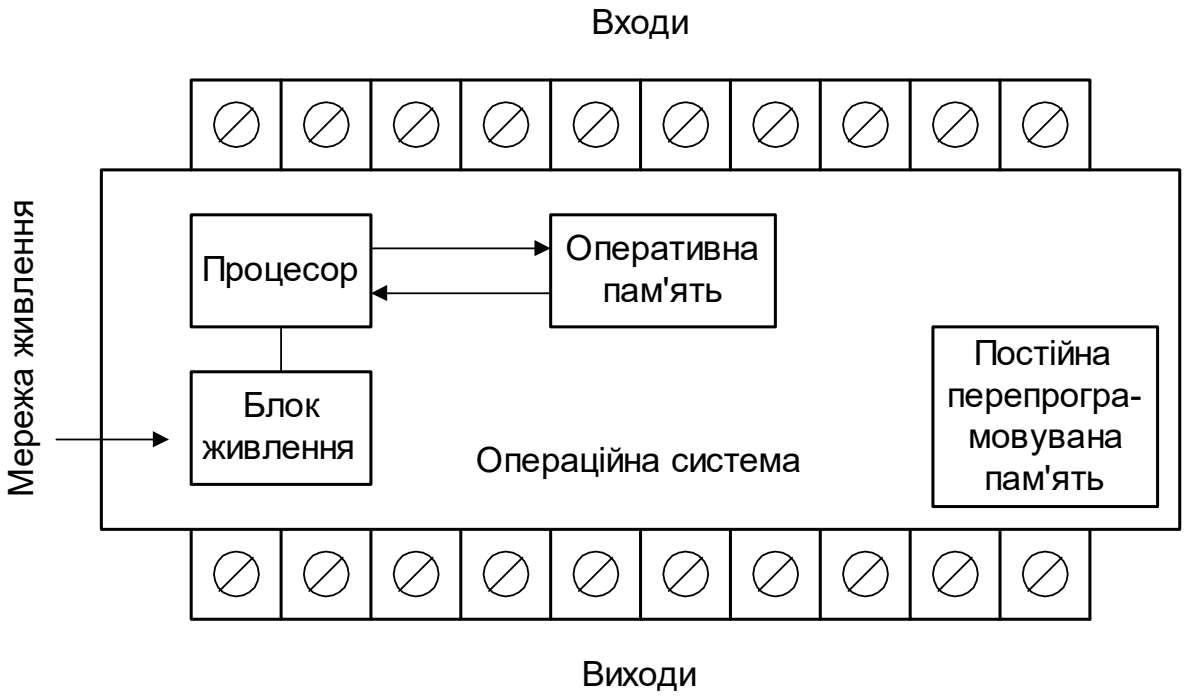


Рисунок 1.3 - Спрощена структура PLC

Виникнення PLC як класу устаткування для промислової автоматизації, продиктоване прагненням користувачів звільнитися від недоліків, властивих дискретним (релейним) пристроям керування з фіксованою логікою. Ця категорія пристроїв промислової автоматики знаходилась в «безпосередньому контакті» з технологічним устаткуванням, тобто на низовому рівні керування технологічними процесами, де власне і стали застосовуватися перші PLC, заміняючи собою (практично один до одного) релейні системи, які існували тоді, дискретно-логічного керування, але вже з можливістю програмування і перепрограмування. Звідси і назва — програмовний логічний контролер. Перший PLC, названий MoDiCon (Modular Digital Controller) був використаний у 1968 р. в автомобільній промисловості США саме для заміни шаф з релейною логікою.

Одночасно з цим, поява мікропроцесорів стимулювала бурхливий розвиток багатопроцесорних і багатомашинних обчислювальних комплексів, що базуються на нових системних інтерфейсах і мережних технологіях. Використовувані спочатку в системах керування військового призначення, багатопроцесорні і багатомашинні комплекси швидко проникнули в промислову сферу, замінюючи собою застарілі на той час мейнфрейми. Нові системи, що одержали назву DCS, на відміну від PLC, розвивалися в напрямку «від загального — до окремого» чи, іншими словами, від загальної, інтегральної постановки задачі автоматизації на всіх рівнях керування технологічними процесами й об'єктами, до створення єдиного розподіленого програмно-технічного середовища для її рішення. Такий розвиток природним чином випливав з необхідності «розподілити» задачі, що раніше виконувалися централізовано на одному мейнфреймі по вузлах розподіленої системи. Перші DCS були використані у виробництвах з неперервним циклом у хімії і нафтопереробці, де, власне, і була необхідність використання великих інтегрованих систем.

Таким чином, історично PLC і DCS були розподілені за типами процесів, що відповідають різним галузям промисловості (рисунок 1.4 і рисунок 1.5).

На сьогоднішній день відмінності DCS від PLC, які вже мають, наприклад, можливість розв’язання задач неперервного регулювання, полягають лише в продуктивності і габаритно-вагових характеристиках, зумовлених історично сформованою різницею підходів до промислової автоматизації «від загального — до окремого» у DCS і «від окремого — до загального» у PLC. Процесорна частина вузлів розподіленої обробки даних низового рівня керування в DCS теж іменується контролером, однак має, як правило, підвищену (у порівнянні з PLC) продуктивністю і більш могутню систему локального введення/виведення. Тут і далі за текстом, говорячи про DCS, будемо розглядати тільки засоби низового рівня контролю і керування, що відповідають PLC за функціональним призначенням і характером розв'язуваних задач. Дійсно, верхній рівень

DCS, як правило, представляє собою програмно-технічні комплекси інформаційної підтримки оператора, які більш чи менш відповідні класичним засобам людино-машинного інтерфейсу і SCADA-пакетам, що традиційно використовуються у комбінації з PLC. Відмінність DCS у цій частині полягає лише в тому, що засоби верхнього рівня контролю і керування технологічними процесами розроблялися, як складова частина всієї системи, а для PLC можна (і зрештою — потрібно) вибирати придатний універсальний SCADA-пакет.

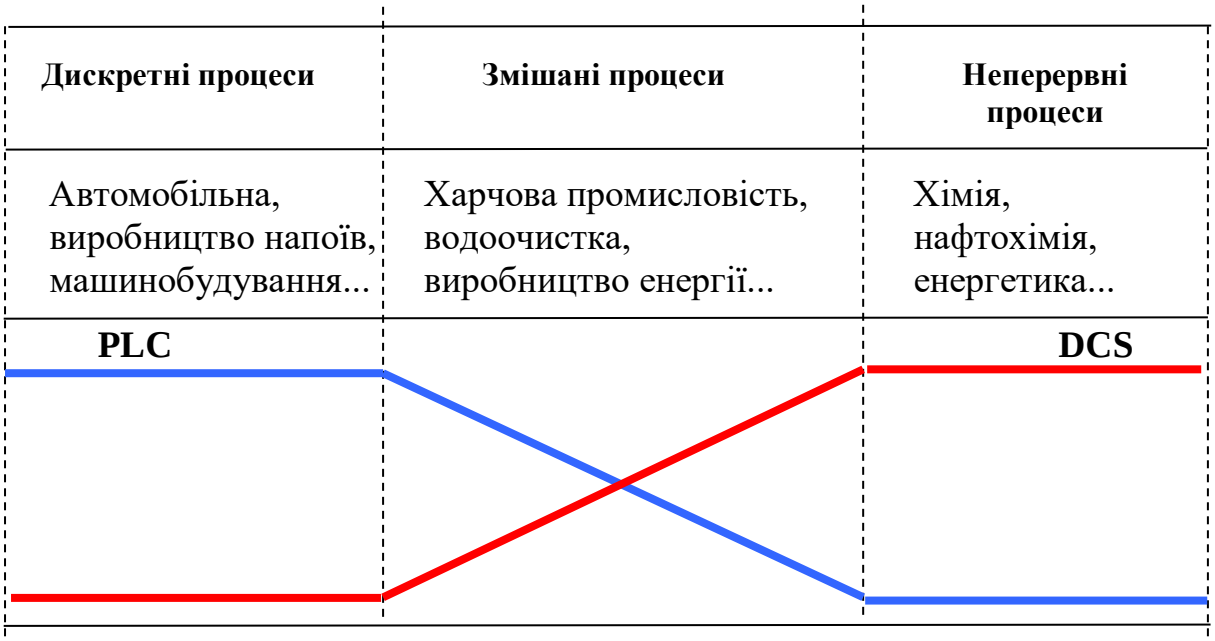


Рисунок 1.4 - Структура процесів у виробництві

В основі типового низового вузла розподіленої обробки DCS лежить, як правило, могутня стандартна магістрально-модульна інтерфейсна система (наприклад VME), розрахована на незалежну одночасну роботу декількох процесорних модулів (контролерів), що мають локальну і загальну розподілену пам'ять. Магістрально-модульна архітектура визначила конструктивне виконання низових вузлів DCS — великі шафи з крейтами для розміщення процесорних модулів, розподіленої загальної пам'яті, мережних інтерфейсів, а також великого числа модулів локального введення/виведення аналогових і дискретних сигналів об'єкта. Наприклад, вузол розподіленої обробки DPU одного з найбільш яскравих і класичних представників розподілених систем керування — сімейства WDPF II фірми «Westinghouse» здатний обслуговувати до 2000 аналогових і дискретних параметрів процесу. Одна чи дві шафи DPU висотою близько 2 метрів можуть бути укомплектовані резервним контролером на системній шині Multibus, 48 чи 96 модулями локального введення/виведення, відповідно, і керувати, наприклад, такими складними технологічними процесами, як подача живильної води і регулювання рівня в парогенераторі атомного енергоблоку ВВЕР-1000 потужністю 1000 МВт.

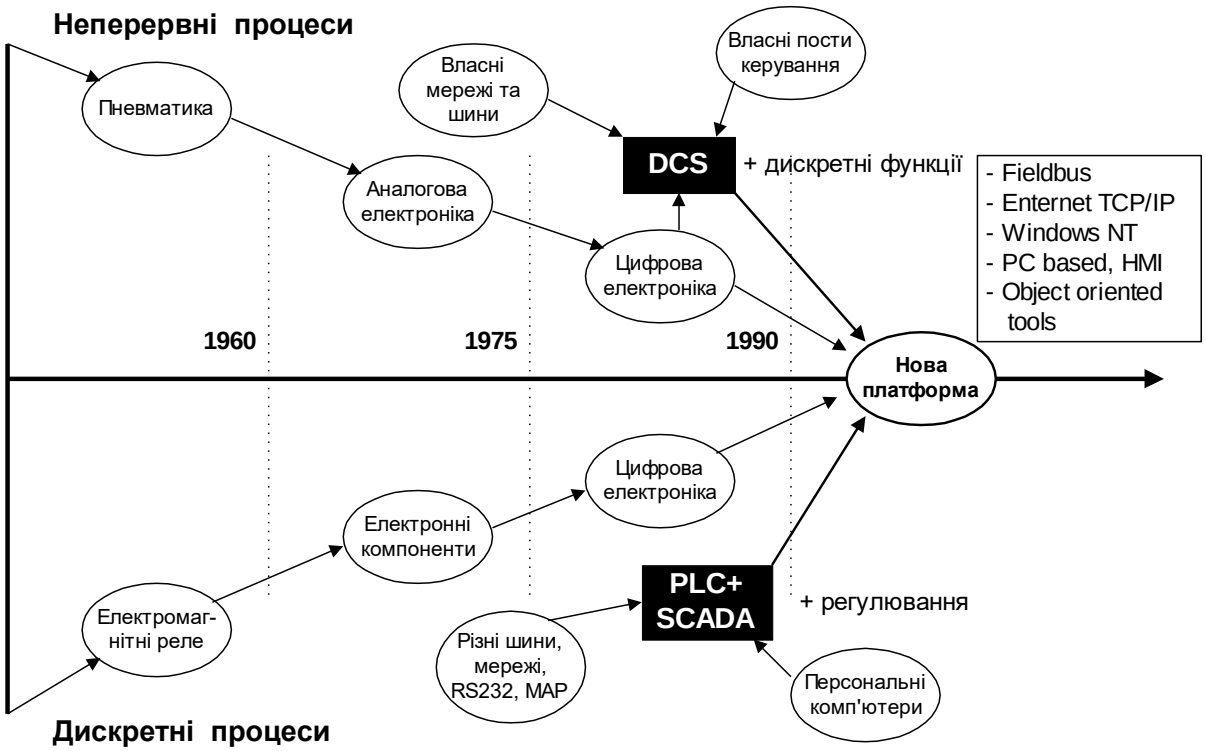


Рисунок 1.5 - Розвиток PLC і DCS

На противагу цьому, найдосконаліші на сьогоднішній день PLC, як правило, мають більш скромні можливості в частині локального введення/виведення, наприклад Modicon TSX Quantum фірми Schneider Electric комплектується не більш, ніж 14 модулями локального введення/виведення на 16-шаровій монтажної панелі з одним процесорним модулем і джерелом живлення. Проте, доповнивши таку структуру засобами розподіленого введення/виведення TSX Quantum, Momentum, можна одержати винятково компактне й економічне технічне рішення для дуже багатьох технологічних процесів на тих же теплових (і навіть атомних) електростанціях, де застосування «монстрів» типу WDPFII чи його наступних модифікацій — Ovation, буде завідомо марнотратним і технічно невиправданим. Розуміючи це, практично усі виробники DCS передбачають у своїй продукції необхідні програмно-технічні засоби зв'язку (інтерфейси) з PLC провідних світових виробників. Так, для зв'язку з PLC Modicon у номенклатурі модулів введення/виведення DPU устаткування WDPF II є спеціалізований інтерфейсний модуль, що реалізує протокол Modbus. Такий же протокол, крім того, є у зв'язувальному сервері WEStation Datalink Server, реалізованому на робочій станції Sun Spare — базовій платформі верхнього рівня розподіленої системи керування WDPF II.

Як уже відзначалося вище, і DCS і їхні малі побратими — PLC спеціально створювалися для вирішення задач керування в реальному

масштабі часу. Саме цим багато в чому обумовлена внутрішня подібність цих двох, що зовні відрізняються, класів устаткування.

Зовнішнім фактором подібності є використання PC як основного засобу візуалізації. В даний час, як основний засіб людино-машинного інтерфейсу і PLC, і DCS використовують персональний комп'ютер (рисунок 1.6).

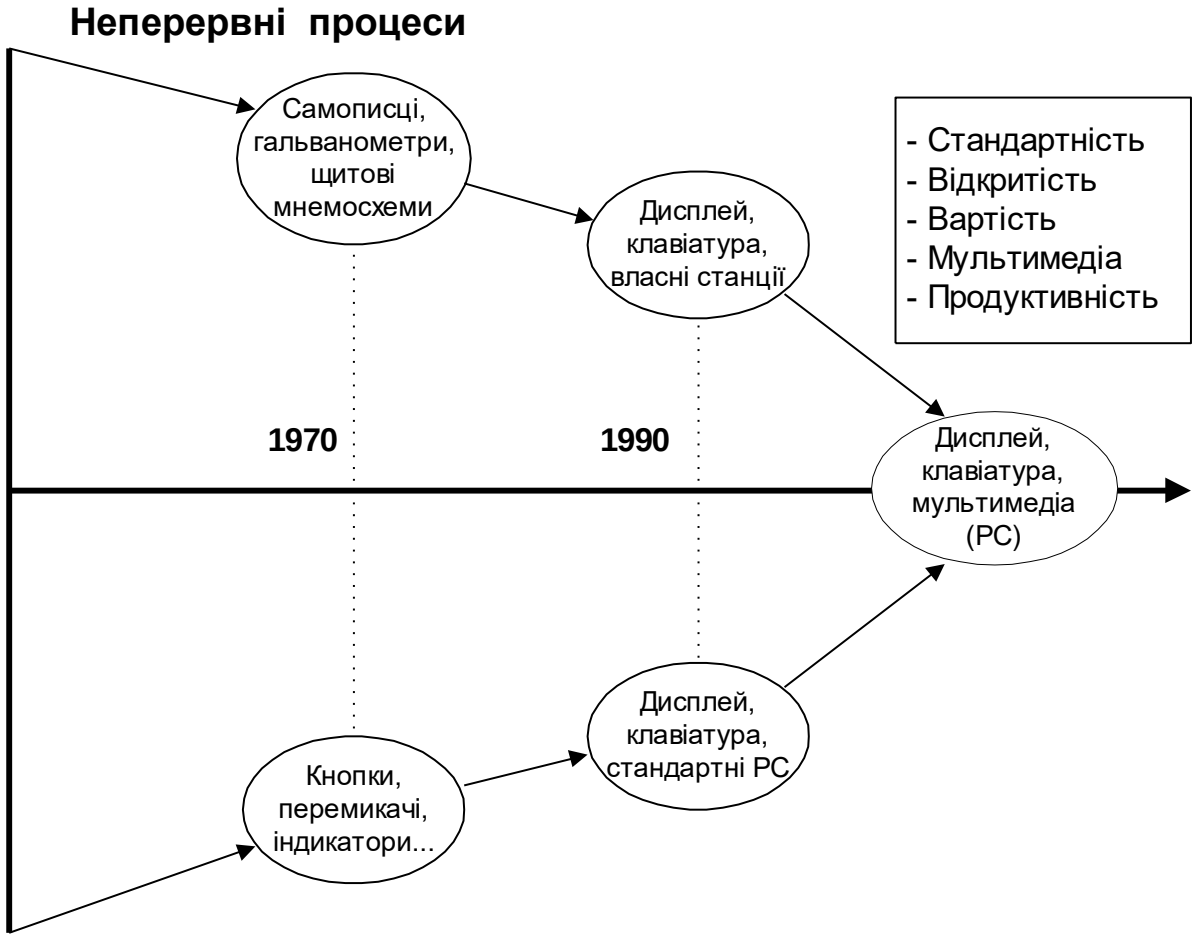


Рисунок 1.6 - Схема розвитку технічних засобів автоматизації

1.3 Організація обчислювального процесу

Націленість на збір та обробку сигналів датчиків, дискретно-логічне керування і неперервне регулювання в реальному масштабі часу визначила архітектуру низових вузлів DCS і PLC. «Вузька» спеціалізація обчислювальної техніки цих двох класів, відсутність периферійних пристроїв (відеомоніторів, принтерів, накопичувачів на гнучких магнітних дисках тощо), характерних для універсальних комп'ютерів, у свою чергу, викликала появу винятково компактних, простих і надійних спеціалізованих операційних систем (ОС) реального часу. Такі ОС зв'язують прикладні програми і дані користувача з апаратурою DCS чи

PLC найбільш природним чином, не страждають «архітектурними надмірностями» і не вимагають спеціальних засобів, що мають місце при спробах адаптувати універсальні комп'ютери до задач промислового контролю і керування (техніка і технології PC based control). Результат — виняткова надійність і «прозорість» організації обчислювального процесу в DCS і PLC, як правило, представляє собою елементарний цикл наступних операцій:

- сканування модулів введення сигналів датчиків об'єкта;
- розв’язання прикладної задачі (задач);
- запис даних у модулі видачі сигналів керування на об'єкт;
- обмін даними по мережі;
- самоконтроль і діагностика.

1.4. Організація введення/виведення

Простуючи до загальної мети з двох різних напрямків, керуючись тими самими вимогами до пристроїв зв'язку з об'єктом автоматизації, розроблювачі DCS і PLC демонструють єдиний підхід до вирішення проблеми введення/виведення аналогових і дискретних сигналів. Архітектура програмно-технічних засобів DCS і PLC надає користувачу можливість організації як **локального** (за місцем розташування контролера DCS чи процесорного модуля PLC), так і **виносного** (за місцем розташування технологічного устаткування об'єкта) введення/виведення.

Локальне введення/виведення (*Local I/O*) використовується у випадках, коли датчики і виконавчі механізми, що беруть участь у реалізації відповідної задачі автоматизації, рівновіддалено розташовані від можливого місця установки контролерної шафи DCS/PLC з модулями введення/виведення, а фактор довжини кабельних ліній зв'язку з об'єктом не є визначальним.

Виносне введення/виведення (*Remote I/O чи просто RIO*) у DCS і PLC будується, як правило, на базі стандартного послідовного інтерфейсу RS-232, RS-422 чи RS-485 і дозволяє рознести групи модулів зв'язку з об'єктом на значну відстань один від одного і від контролерної/процесорної частини (від декількох десятків метрів до кілометра і більше).

Обидва варіанти можуть комбінуватися один з одним для одержання оптимальної конфігурації введення/виведення під конкретне розташування об'єкта автоматизації і вимоги до розв'язуваних задач.

1.5 Організація інформаційного обміну і розподіленої обробки даних

Засоби інформаційного обміну між вузлами, що забезпечують розподілену обробку даних, з'явилися на світ як невід'ємна частина єдиної інтегрованої концепції DCS. На противагу цьому, розподілена обробка даних на PLC, як і можливість роботи з аналоговими параметрами процесу, виникла не відразу, а з'явилася закономірним логічним підсумком розвитку цієї техніки «від окремого — до загального»: від простих окремих контролерів, що автономно керують конкретним промисловим устаткуванням, до складних, розгалужених структур з великим числом PLC, що координовано контролюють складні технологічні процеси.

В основі розподіленої обробки інформації і керування, реалізованих у DCS і найкращих PLC, лежить структурована комунікаційна система на базі будь-якої польової шини (магістралі даних реального часу). Розглянемо основні властивості двох представників комунікаційних систем для розподіленого керування: магістралі даних Westnet II сімейства DCS WDPF II і польової шини Modbus Plus програмовних контролерів серії Modicon TSX. Магістраль Westnet II підтримує до 254 вузлів у мережі і гарантоване відновлення до 16 000 параметрів процесу раз у секунду при фіксованій швидкості передачі 2 Мбіт/с. Modbus Plus працює на швидкості 1 Мбіт/с і дозволяє поєднувати в розподілену систему керування до 32-х, а при використанні повторювача — до 64-х вузлів PLC. Обидві ці системи реалізують принцип рівноправного безконфліктного доступу усіх вузлів до загальної шини через механізм послідовної передачі чи маркера «права володіння» магістраллю від вузла до вузла по так званому логічному кільцю. Поточний власник маркера передає дані в магістраль, а всі інші вузли споживають їх за необхідністю. Закон циркуляції маркера в логічному кільці розподіленої системи з усією необхідною обробкою аварійних ситуацій розміщений у мережному інтерфейсному модулі кожного вузла, завдяки чому ні Westnet II, ні Modbus Plus не вимагають наявності в мережі будь-яких елементів централізованого керування передачею даних (майстрів-вузлів, моніторів і т.п.). Детермінований характер протоколу обміну і характеристик пропускної здатності в цих двох системах дозволяє розробнику АСУ ТП з 100%-ю вірогідністю оцінити період відновлення інформації і можливу реакцію вузлів на ті чи інші події. Ця властивість комунікаційних систем визначила їхнє широке застосування в АСУ ТП неперервними процесами, де особливе значення має час реакції розподілених контурів регулювання на зміну вхідних даних, що надходять від датчиків і суміжних підсистем.

1.6 Інструментарій для створення програмного забезпечення

Останній елемент порівняння двох розглянутих технологій — інструментальні засоби створення прикладного програмного забезпечення задач збору й обробки інформації датчиків, дискретно-логічного керування і регулювання параметрів технологічного процесу. Для програмування DCS і PLC давно використовуються графічні мови високого рівня, що не вимагають від користувача кваліфікації класичного програміста.

Провідні виробники програмовних логічних контролерів сьогодні орієнтуються на міжнародний стандарт IEC-1131-3, що встановлює вимоги до п'яти основних мов програмування PLC, серед яких, наприклад, великою популярністю користується LD (Ladder Diagram) чи мова релейно-контактних схем. Суть цієї мови — у представленні логіки функціонування системи керування у вигляді добре зрозумілих будь-якому професійному електрику розгорнутих схем з'єднань «контактів» і «обмоток реле», прив'язаних до вхідних, вихідних і проміжних змінних створюваної програми. Так само близькі і зрозумілі професіоналам в області автоматичного керування функціональні схеми законів регулювання, покладені в основу іншої популярної графічної мови — FBD (Function Block Diagram).

Елементи цих і інших, широко відомих форм представлення прикладних програм дискретно-логічного і неперервного керування давно використовуються й у DCS. Прикладом може бути графічний пакет Control Builder для програмування низових вузлів розподіленої обробки сімейств WDPF II і Ovation. Цей пакет, хоча і не базується на стандарті IEC-1131-3, має багато загального зі згаданою вище мовою FBD.

1.7 Сфери використання PLC і DCS

Досить часто замовники і користувачі АСУ неперервними ТП задаються важливим для виробничника питанням: що краще — DCS чи PLC? Мабуть немає однозначної відповіді на це питання без детального розгляду всіх умов і обставин, що мають місце при вирішенні конкретних задач автоматизації. Проте, деякі рекомендації, сформульовані в загальному вигляді, можуть дещо допомогти зацікавленим фахівцям на ранніх стадіях розробки проектів:

1. Якщо Ви маєте справу з великою кількістю вхідних/вихідних сигналів об'єкта (від декількох тисяч і більше), а сам об'єкт являє собою відносно компактно розташовану сукупність тісно взаємозалежних технологічних систем (наприклад — енергоблок теплової чи атомної електростанції), є сенс придивитися до DCS і зробити порівняльну вартісну оцінку проекту стосовно варіанта на основі PLC.

2. Якщо Вам потрібно здійснювати диспетчерський контроль і керування великим числом територіально розосереджених невеликих автономних об'єктів (наприклад — міська тепломережа, електропідстанції і розподільні мережі, газо- і нафтопроводи), то перевагу варто віддати системам телемеханіки на основі PLC.

3. Якщо перед вами - масштабна задача інтегральної автоматизації об'єктів першого і другого типу, то було б не зайвим подумати і про комбінований варіант — DCS плюс PLC.

1.8 Загальна характеристика гами PLC TSX Modicon

У гамі представлені кілька контролерів для вирішення задач різного ступеня складності:

- мініатюрний **Modicon TSX Nano** для вирішення найпростіших задач автоматизації невеликих машин і установок;
- **Modicon TSX Micro** для вирішення задач середньої складності;
- **Modicon TSX Premium** для вирішення складних задач;
- **Modicon TSX Quantum** для вирішення найбільш складних задач у великих системах керування;
- **Modicon TSX Compact**, який найбільш придатний для задач SCADA/RTU;
- **Modicon TSX Momentum** для задач економічного розподілу введення/виведення і незалежної обробки даних.

Якщо необхідно створити систему керування невеликою машиною чи механізмом, то тут незамінним виявиться **Modicon TSX Nano** (рисунок 1.7), побудований за моноблочним принципом. Незважаючи на невеликі габарити, у цьому PLC є все необхідне для побудови повноцінної системи керування:

- процесор, здатний виконувати 1000 логічних інструкцій за 200 мкс;
- пам'ять, достатня для розміщення 1К інструкцій;
- 16 годинників астрономічного часу;
- лічильники, таймери і т.д.

У залежності від виконання, TSX Nano оснащується джерелом живлення на 24В постійного чи 100-240В змінного струму. Можуть бути обрані входи 24В постійного чи 115В змінного струму, а також транзисторні (0.5А) чи релейні (2А) виходи. Доступний у трьох типорозмірах (рисунок 1.8), цей PLC забезпечує "мінімально необхідне" вирішення для вимог конкретного використання. Можуть бути зібрані конфігурації з кількістю входів/виходів від 10 до 48 (з урахуванням розширення). Є спеціалізований лічильний вхід (лічильна частота до 10КГц), який можна використовувати як частотомір і тригерні входи (з

мінімальним часом включення 100 мкс). Картину доповнюють 2 комунікаційних порти для включення TSX Nano у мережні архітектури зв'язку з пристроями низової автоматики, операторськими панелями. Можуть бути використані протоколи MODBUS, UNI-TELWAY чи ASCII.

Рисунок 1.7 - Загальний вигляд TSX Nano

Рисунок 1.8 - Варіанти TSX Nano

Програмування TSX Nano виконується за допомогою ручного пульта FTX 117 (рисунок 1.9) чи програмного забезпечення PL7-07 (під DOS) на персональному комп'ютері на мовах стандарту МЕК 1131-3 (Релейно-контактні Схеми і Список Інструкцій). Якщо врахувати дуже низьку вартість TSX Nano, то виходить привабливий контролер для рішення простих задач і заміни схем релейної автоматики.

Інший представник гами - **Modicon TSX Micro** (рисунок 1.10) спроектований для задоволення вимог до простоти створення систем керування і легкості технічного обслуговування. Його модульність і компактність дають можливість створювати недорогі рішення для систем автоматизації не тільки простим устаткуванням, але і більш складним (з кількістю входів-виходів до 248).

Обчислювальна потужність TSX Micro, гідна самих могутніх контролерів, здатна задовольнити вимоги до часу реакції найвисокопродуктивнішого устаткування, а гама модулів введення/виведення і вбудованих прикладних функцій дозволяє охоплювати найширші області застосування: швидкий рахунок, керування

позиціонуванням (від 500 Гц до 40 КГц), вимірювання і ПІД-регулювання, високошвидкісна дискретна обробка.

Рисунок 1.9 - Загальний вигляд пульта FTX 117

Рисунок 1.10 - Загальний вигляд TSX Micro

Широкі комунікаційні можливості цього контролера дають можливість легко підключатися як до польових шин (Uni-telway, Modbus,

Fipio), так і до мереж (Fipway, Modbus Plus). Стикування з засобами людино-машинного інтерфейсу також не викликають проблем. Є цілий ряд рішень, що значно знижують вартість монтажу і підключення.

Віддалені термінальні блоки (клемники) TELEFAST (рисунок 1.11) дозволяють просто реалізувати функції захисту і поділу каналів, відображення їхнього стану, а також перетворення сигналу. Шина датчиків As-i представляє собою двопровідну лінію на якій можна підключати до 128 дискретних датчиків і виконавчих механізмів на відстані до 100 метрів. До шини As-i пропонується широка гама пристроїв, як від Schneider Electric, так і від інших виробників. Підтримуються також звичайні датчики і приводи (пускачі) дискретного типу.

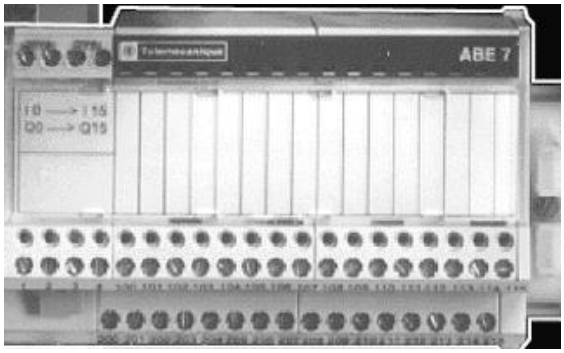


Рисунок 1.11 - Клемник TELEFAST

Базова конфігурація TSX Micro містить у собі:

- шасі (з 2-ма чи 3-ма слотами для установки модулів введення-виведення);
- джерело живлення (змінний чи постійний струми);
- процесор і система резервного копіювання.

Вбудований діагностичний дисплейний модуль дозволяє відображення стану каналів і модулів введення-виведення, їхню діагностику і навіть стан внутрішніх змінних контролера. Перший рівень обслуговування PLC - визначення стану плюс виявлення і локалізація несправностей - проводиться без усяких додаткових засобів.

TSX Micro доступний у трьох варіантах базових конфігурацій (рисунок 1.12):

- компактна (2 слоти розширення) з передустановленим модулем введення-виведення;
- конфігурованим (3 слоти) із двома портами PCMCIA для розширення пам'яті й установки комунікаційних карт;
- розширена конфігурована (3 слоти/2 порти PCMCIA) з додатковими інтегрованими аналоговими входами-виходами (8 входів-1 вихід) і функціями швидкого рахунку: два канали 10 КГц.

Рисунок 1.12 - Варіанти TSX Micro

Завдяки всім перерахованим вище достоїнствам, TSX Micro ідеальний для створення систем керування сучасними машинами і технологічним устаткуванням, у тому числі і мобільними.

Але якщо задача по-справжньому складна, то тут не обійтися без більш могутнього PLC, представленого в гамі - **Modicon TSX Premium** (рисунок 1.13). Цей контролер призначений для побудови складних розподілених систем керування і ґрунтується на архітектурі X-BUS, так званої мережі станцій реального часу. Ця архітектура дозволяє розподіляти усі функції керування на відстань до 500 метрів і при цьому для вилучених підсистем гарантується той же час реакції, що і для локальних, тобто близько 1.5 мс. Є дві основні переваги цього вирішення:

- вільний розподіл функцій на усій відстані без всяких обмежень;
- значне скорочення витрат на систему керування за рахунок зниження витрат на кабельну продукцію (модулі введення/виведення встановлюються якнайближче до оперативної частини) і на комунікаційні співпроцесори (до 100 м вони відсутні).

Рисунок 1.13 - Загальний вигляд TSX Premium

Schneider Automation є піонером у реалізації подібних архітектур. Але X-BUS не єдине достоїнство цього контролера. Крім класичних функцій, таких як дискретне і аналогове введення/виведення інформації та великі комунікаційні можливості, у ньому втілений цілий ряд рішень високого рівня. Це гама модулів швидкого рахунку (аж до 250 КГц), модулі керування позиціонуванням і кроковими двигунами. Є спеціальні модулі і для вимірювання ваги.

Надзвичайно широкий спектр підтримуваних TSX Premium мереж. Крім мереж, підтримуваних TSX Micro, TSX Premium підтримує ще і ETHERNET TCP/IP, ETHWAY, ETHERNET MMS, PROFIBUS FMS, INTERBUS-S (ведучий), PROFIBUS-DP. Така палітра дозволяє легко вбудовувати TSX Premium у вже наявні системи керування, у тому числі, створених на устаткуванні інших виробників.

Для Premium доступно 10 типів процесорних модулів, що відрізняються по швидкодії, кількості підтримуваних дискретних і аналогових входів/виходів, орієнтації на спецфункції (регулювання) і, природно, за ціною. Є великий набір модулів дискретного й аналогового введення/виведення з різною кількістю каналів, характеристиками, різні шасі і джерела живлення. Така розмаїтість пристроїв дозволяє будувати

системи керування мінімально достатні для даного використання й оптимальні за ціною.

Найбільш потужним PLC є **TSX Quantum** (рисунок 1.14), який використовують у великих системах керування. Даний контролер реалізований на основі процесорів Intel 486/586 і має цілу гаму конфігурованих модулів введення/виведення: дискретне введення/ виведення із щільністю 96 каналів на модуль; аналогове введення/ виведення – до 32 каналів на модуль; мережеві модулі.

Рисунок 1.14 - Загальний вигляд TSX Quantum

Найбільш придатним для вирішення задач SCADA/RTU є контролер **Modicon TSX Compact** (рисунок 1.15). У порівнянні з іншими контролерами Schneider Automation він має такі переваги:

- робота в режимі температур - 40°C до + 70°C;
- спеціальне покриття модулів для роботи в хімічно агресивних середовищах;
- унікальні функції комунікації - підтримка протоколів Modbus (який є де-факто стандартом для RTU), Modsus Plus, Interbus S. За протоколом Modbus контролер Compact може бути включений у глобальні мережі (WAN) через радіомодемний, телефонний або супутниковий зв'язок. Крім того, завдяки функціональному блоку XMIT віддалений PLC-slave може бути перетворений у PLC-master і, таким чином, набагато швидше сповіщати центральний пост SCADA про будь-які події на об'єкті;
- вбудовані алгоритми обчислення потоків рідких речовин і газів, що відповідають стандартам AGA 3&8 (American Gaz Association) і ISO 12213;

- велика пам'ять (з розширенням - до 4 MB у форматі PCMCIA карти) для локального збереження даних;
- наявність іскро-вибухо-безпечних каналів введення/виведення;
- наявність комплексів “готових до використання”, для вирішення ряду задач в областях водо-, теплопостачання, транспорту нафти і газу й в електроенергетиці. Ці вирішення, - Pump-Pak і RTU-Pak (виробництво Square D), представляють собою встановлений у шафі Compact з операторським дисплеєм Magelis і передпрограмованою для даної задачі логікою.

Рисунок 1.15 - Загальний вигляд TSX Compact

TSX Momentum (рисунок 1.16) - це функціонально повне сімейство засобів керування - модулів розподіленого введення/виведення, процесорів, комунікаційних адаптерів і адаптерів розширення. Даний процесор має такі особливості:

- Економічне розподілене введення/виведення

Для використань, які вимагають територіально віддаленого розміщення "вузлів" (концентраторів) введення/виведення, TSX Momentum (зв'язкові адаптери і базові пристрої введення/виведення - ПВВ) забезпечує економічно ефективне розташування ПВВ у безпосередній близькості від технологічного устаткування з використанням польових шин багатьох розповсюджених відкритих промислових мереж. Наприклад, може знадобитися розгортання розподіленого введення/виведення в системі керування на базі персонального комп'ютера. Незалежно від того, чи має персональний комп'ютер інтерфейс Modbus Plus, FIPIO, Ethernet, Interbus, Profibus, DeviceNet, чи ControlNet, TSX Momentum здатний забезпечити стабільність технічних рішень.

Рисунок 1.16 - Загальний вигляд TSX Momentum

- Незалежна обробка

Є можливість поставити процесорний адаптер на базовому ПВВ замість комунікаційного. Маючи широкий вибір базових ПВВ, на яких встановлюється процесорний адаптер, можна отримати простий і економічний доступ до великого числа використань низового рівня керування. Якщо потрібна самостійна система локального керування з числом параметрів від 16 до 32, є базові ПВВ із комбінованим введенням/виведенням дискретних сигналів постійного і змінного струму. Для випадків, коли потрібна підвищена швидкість обробки, існують швидкісні ПВВ, що можуть використовуватися разом із швидкодійним процесорним адаптером FastScan M1. І, якщо вимагаються аналогові функції, є багатофункціональний базовий ПВВ, що має чотири аналогових входи, два аналогових виходи і шість дискретних (4 вхідних і 2 вихідних) сигналів 24 В постійного струму. На додаток до стандартного порту Modbus для програмування, деякі процесорні адаптери мають другий порт, що підтримує розподілене введення/виведення. Використовуючи комунікаційні адаптери Interbus, можна розосередити до 80 модулів TSX Momentum на відстані до 8 миль від одного процесорного адаптера і базового ПВВ.

- Розподілені системи

У більш великих, інтегрованих керувальних структурах, TSX Momentum може розвантажувати, спрощувати, зв'язувати, розподіляти, чи

консолідувати, усувати недоліки традиційних систем. TSX Momentum розширює архітектуру Modicon TSX Quantum і TSX Compact.

Список питань для самоконтролю

1. Дайте визначення поняттям DCS, PLC, PC based control, SCADA, SCADA Host, MTU, RTU.
2. Наведіть спрощену структуру DCS.
3. Наведіть спрощену структуру PLC.
4. Який розподіл DCS і PLC у системах керування?
5. Які основні можливості систем керування на базі DCS і PLC?
6. Як організується обчислювальний процес у системах керування?
7. Як організується введення/виведення у системах керування?
8. Як організується інформаційний обмін і розподілена обробка даних у системах керування?
9. Який інструментарій використовується для створення програмного забезпечення у системах керування?
10. Яка сфера використання DCS і PLC?
11. Дайте загальну характеристику гами PLC TSX Modicon.
12. Які основні можливості PLC TSX Nano?
13. Які основні можливості PLC TSX Micro?
14. Які основні можливості PLC TSX Premium?
15. Які основні можливості PLC TSX Quantum?
16. Які основні можливості PLC TSX Compact?
17. Які основні можливості PLC TSX Momentum?

2 БУДОВА ПЛК MODICON TSX MICRO

2.1 Основні характеристики TSX Micro

Для кращої відповідності вимогам користувача ПЛК TSX Micro (рисунок 2.1) випускається у трьох виконаннях:

TSX 37-10 - компактний, модульний ПЛК, що випускається в шести базових конфігураціях, які відрізняються напругою живлення і типом дискретного модуля входів-виходів, встановленого в першому слоті. Максимальна кількість входів-виходів для цих типів ПЛК складає: 128 - при використанні з'єднань "під гвинт", і 184 - при використанні з'єднувачів HE 10.

TSX 37-21 - модульний, з вмонтованим астрономічним годинником, з можливістю розширення розміру пам'яті й встановлення комунікаційного модуля. Цей ПЛК не має вмонтованих входів-виходів, але дозволяє встановлювати до 160 входів-виходів при використанні з'єднань "під гвинт" і до 248 - при використанні з'єднувачів HE 10. ПЛК TSX 37-21 мають 2 конфігурації, які відрізняються напругою живлення змінного (AC) або постійного (DC) струму.

TSX 37-22 модульний, має характеристики, ідентичні ПЛК TSX 37-21, але в ньому є вмонтовані аналогові входи-виходи і лічильні канали.

Всі типи дискретних і аналогових модулів, модулів швидкого рахування та ін. можуть бути встановлені у всі доступні слоти ПЛК. Для кращої адаптації до вимог користувача дискретні модулі бувають двох форматів: стандартний, що займає один слот (2 місця для встановлення) і напівформатний, який займає тільки одне місце для встановлення. Всі інші модулі (аналогові, розрахункові й ін.) - напівформатні. За допомогою міні-шасі розширення, що може бути безпосередньо підімкнене до базового шасі, можна збільшити кількість доступних слотів і, відповідно, кількість модулів, що встановлюються.

2.2 Дискретні модулі входів-виходів

Дискретні модулі входів-виходів (I/O) відрізняються не тільки форматом (стандартні і напівформатні), але і кількістю каналів (від 4 виходів до 64 I/O), типом входів (постійного - DC або змінного струму - AC), типом виходів (транзисторні або релейні) і підключенням (гвинтова клемна колодка або HE10 з'єднання).

Модулі постійного струму (24 V DC) поставляються як із з'єднанням "під гвинт", так і з з'єднувачами HE 10; за винятком вихідного модуля 24 V DC/2A і модулів на 32 входи і 32 виходи, які мають тільки підключення

"під гвинт" і модуля високої щільності на 64 входи-виходи, який має тільки НЕ 10 з'єднання.

Інші модулі, включаючи модулі змінного струму AC і-або релейні виходи завжди обладнані гвинтовою клемною колодкою.

Як віддалені входи-виходи можна використати до 4 TSX Nano, що дозволяє збільшити відстань до входів-виходів (до 200 метрів) і збільшити їх кількість у конфігурації (рисунок 2.4).

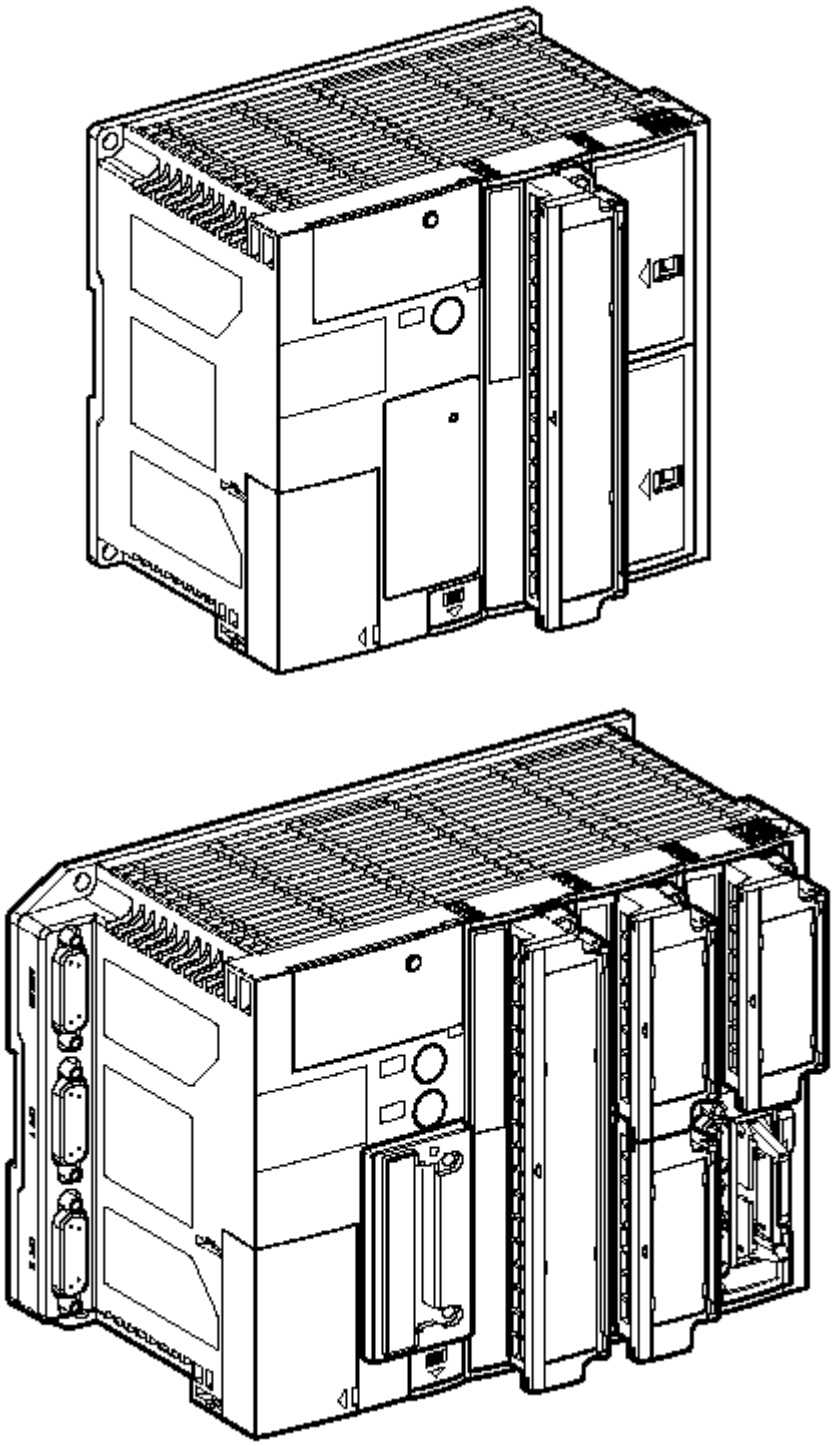


Рисунок 2.1 - Загальний вигляд TSX Micro

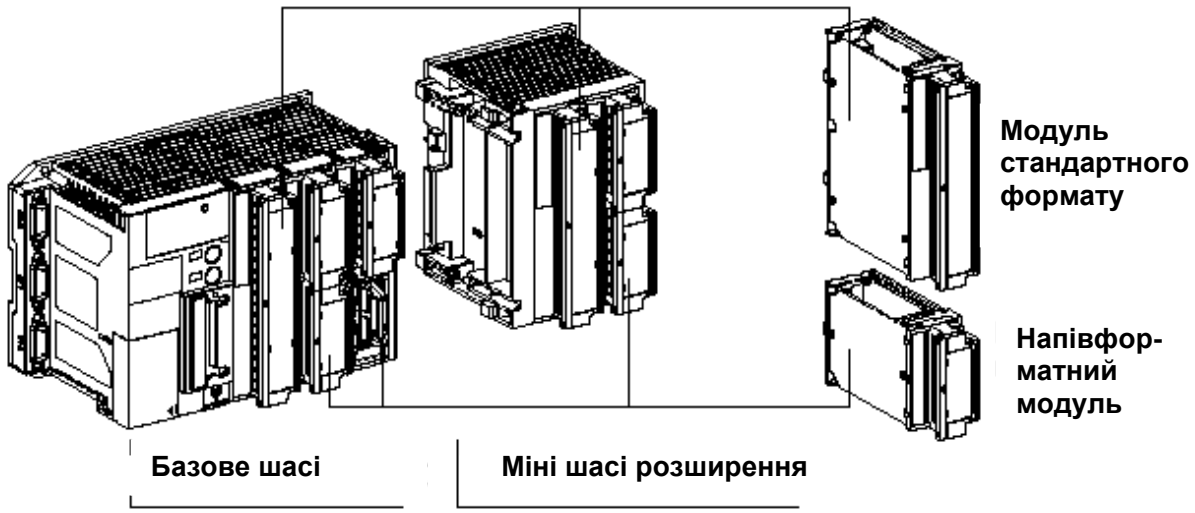


Рисунок 2.2 - Шасі та модулі TSX Micro

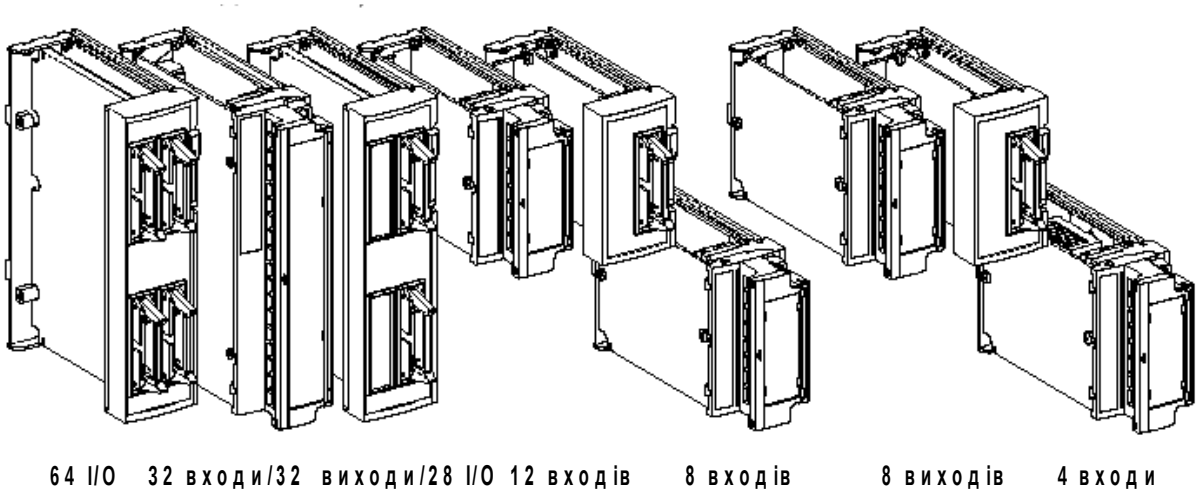


Рисунок 2.3 - Дискретні модулі TSX Micro

2.3 Аналогові входи-виходи

Аналогові модулі TSX Micro відрізняються кількістю каналів, їх характеристиками і діапазоном вимірювань (рівень напруги, термопари, термометри опору та ін.).

- ПЛК TSX 37-22 має 8 аналогових 0-10 В 8-бітових входів і 1 аналоговий 0-10 В 8-бітовий вихід, еталонний 10 В вихід (рисунок 2.5), забезпечуючи економічне рішення для цілого ряду прикладних задач. Ці входи можуть бути використані разом із модулем коригування й адаптації, що дозволяє:
- ручне регулювання змінних програми за допомогою 4-х потенціометрів;
- перетворення сигналу 4-20 мА в 0-10 В;

- підключення дискретних сигналів 24 В до аналогових входів.

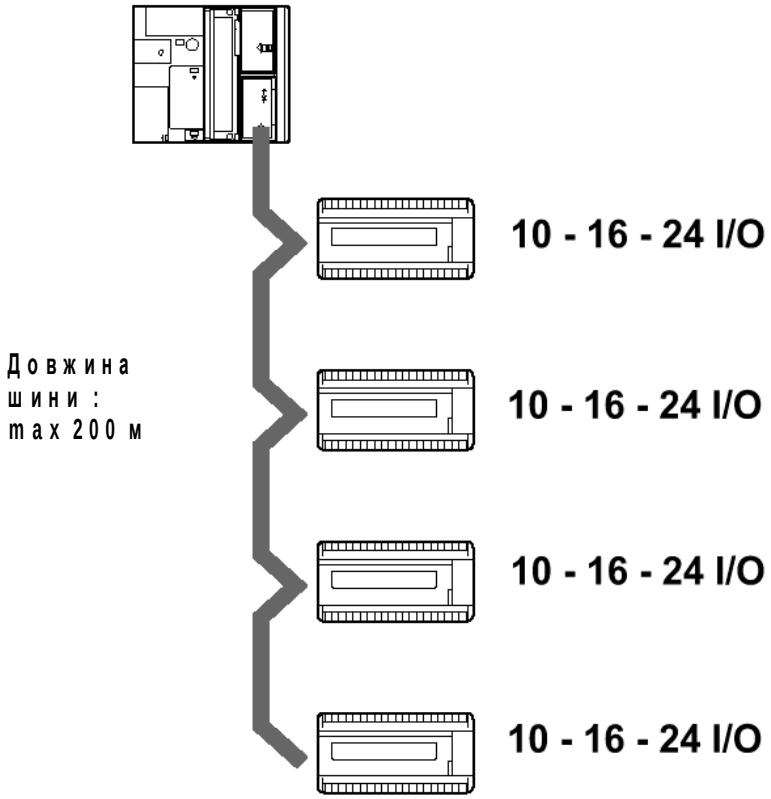


Рисунок 2.4 - Використання TSX Nano як віддалених дискретних входів-виходів

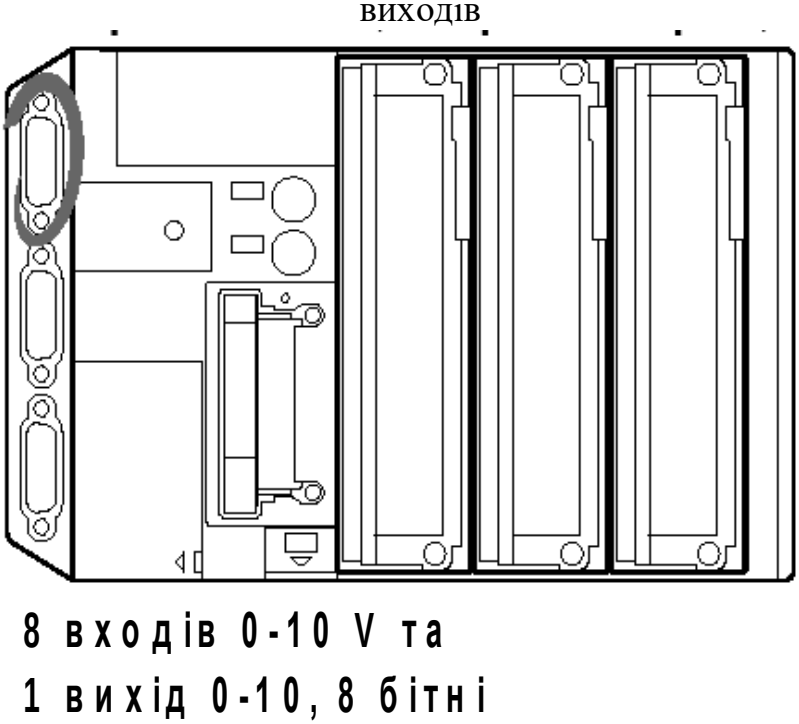


Рисунок 2.5 - Вбудовані аналогові модулі ПЛК TSX 37-22

Модулі аналогових входів і виходів (рисунок 2.6) можуть бути встановлені у всі типи ПЛК, забезпечуючи високу точність. Вони відрізняються за кількостю каналів (від 2 до 8) і типом входів-виходів (рівень напруги або струму, термомодулі та ін.). Підключення проводів від сенсорів здійснюється за допомогою гвинтової клемної колодки.

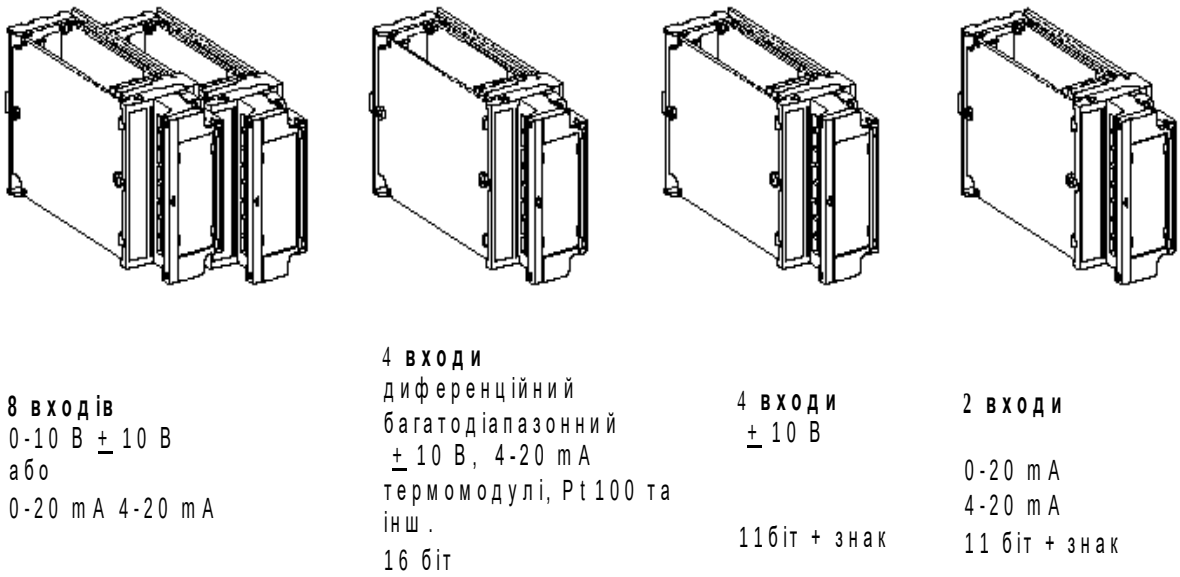


Рисунок 2.6 - Модулі аналогових входів і виходів TSX Micro

2.4 Лічильні канали

ПЛК TSX Micro дозволяє використовувати три методи рахування імпульсів:

• з використанням вмонтованих у ПЛК TSX 37-22 лічильних каналів;

• з використанням дискретних входів першого модуля;

• з використанням спеціалізованих лічильних модулів (TSX CTZ 1A/2A, TSX CTZ 2AA), що можуть бути встановлені у доступний слот.

Перші чотири входи дискретного модуля, встановленого в першому слоті ПЛК, забезпечують два 500 Гц реверсивні лічильні канали (рисунок 2.7).

У ПЛК TSX 37-22 вмонтовані два накопичувальних 10 кГц лічильні канали (рисунок 2.8), які також укомплектовані функціями скидання, попереднього встановлення й занулення лічильників.

Лічильні модулі (прямі, зворотні, реверсивні лічильники) відрізняються за кількостю каналів, граничною частотою (40кГц або 500кГц), типом і кількістю відповідних логічних сигналів (рисунок 2.9).

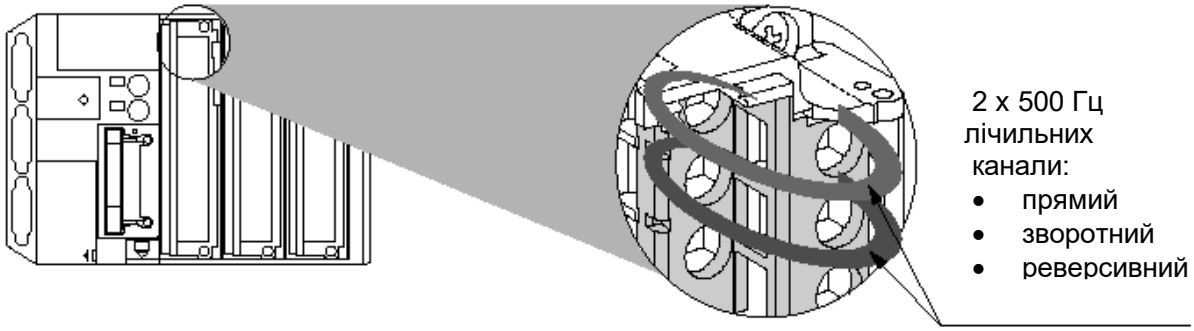
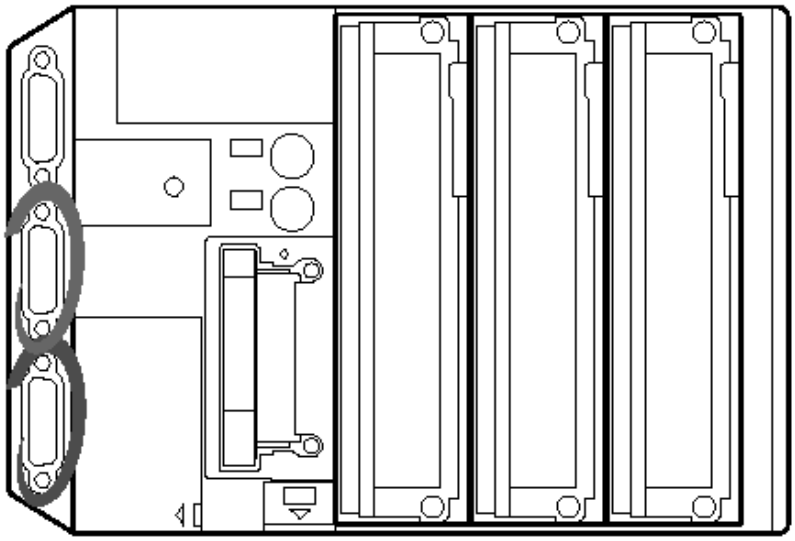


Рисунок 2.7 - Лічильні канали 2×500 Гц



2 x 10 кГц лічильні канали:

- прямий,
- зворотний,
- реверсивний (на першому каналі)

Рисунок 2.8 - Накопичувальні 2×10 кГц лічильні канали

2.5 Комунікаційні можливості TSX Micro

ПЛК TSX Micro легко підключається до багатоточкових послідовних каналів передачі даних за допомогою термінального порту (для всіх моделей ПЛК) і додаткового порту для підключення засобів людино-машинного інтерфейсу (для ПЛК TSX 37-21/22). Ці порти дозволяють підключити (тільки по одному протоколу одночасно):

- програмний термінал і-або людино-машинний інтерфейс (режим головного UNI-TELWAY),

- багатоточковий зв'язок ПЛК по мережі UNI-TELWAY (UNI-TELWAY головний-підлеглий),
- принтер або термінал у символьному режимі,
- модем.

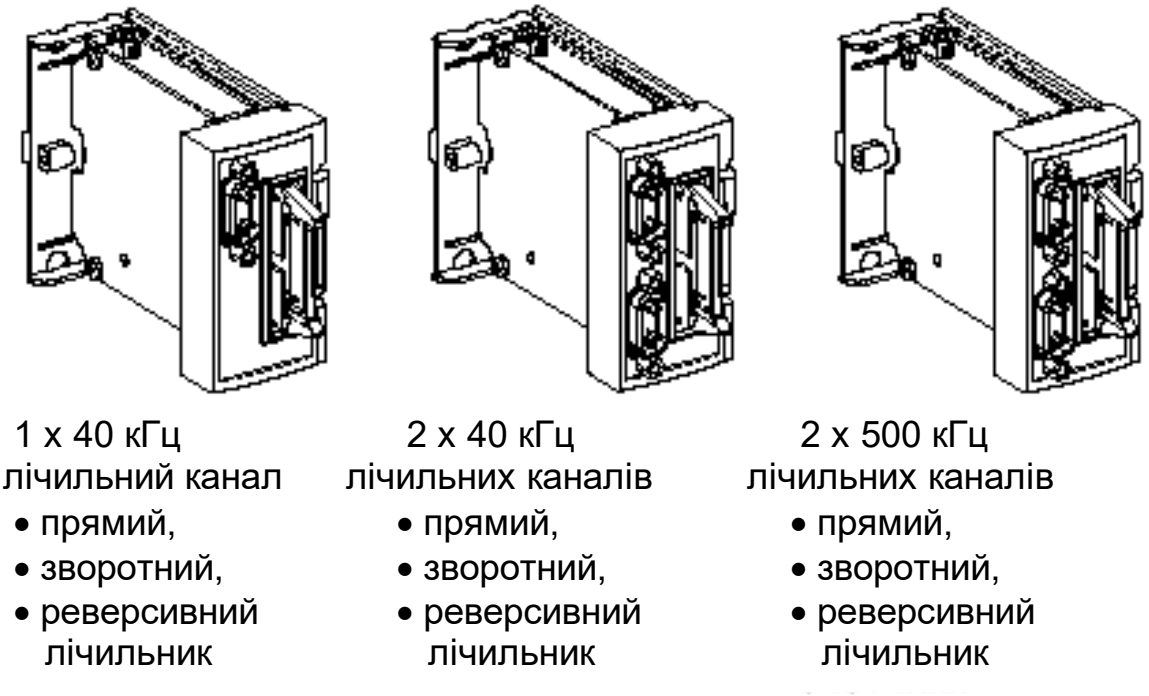


Рисунок 2.9 - Лічильні модулі TSX Micro

Ізолювальний пристрій TSX P ACC 01 дозволяє підключати ПЛК до шини UNI-TELWAY, коли відстань перевищує 10 метрів. Додатково він дублює термінальний порт, що дозволяє підключити до ПЛК TSX 37-10 одночасно програмний термінал і засоби людино-машинного інтерфейсу (рисунок 2.10).

ПЛК TSX 37-21 і TSX 37-22 обладнані слотом для комунікаційної карти формату PCMCIA (дуплексний або напівдуплексний послідовний асинхронний зв'язок, UNI-TELWAY, JBUS/MODBUS, FIPWAY, Agent FIPIO і Modbus+) (рисунок 2.11).

2.6. Примусова вентиляція ПЛК

В залежності від типу ПЛК (TSX 37 10 або TSX 37 21/22 з/або без міні-шасі розширення), один або два вентиляційних модуля можуть бути встановлені над кожним ПЛК, для того щоб примусовим засобом охолоджувати різноманітні модулі. Ці вентилятори (рисунок 2.12) повинні використовуватися в таких випадках:

• Навколишня температура в діапазоні 25°C...60°C: примусова вентиляція збільшує термін служби різноманітних компонентів ПЛК TSX Micro (збільшення 25% у MTBF).

• Навколишня температура в діапазоні 60°C...70°C: оскільки температура без вентиляції обмежена 60°C, примусова вентиляція використовується для зменшення температури всередині модулів більше ніж на 10°C і приведення внутрішньої температури модулів до еквівалента навколишнього середовища 60°C.

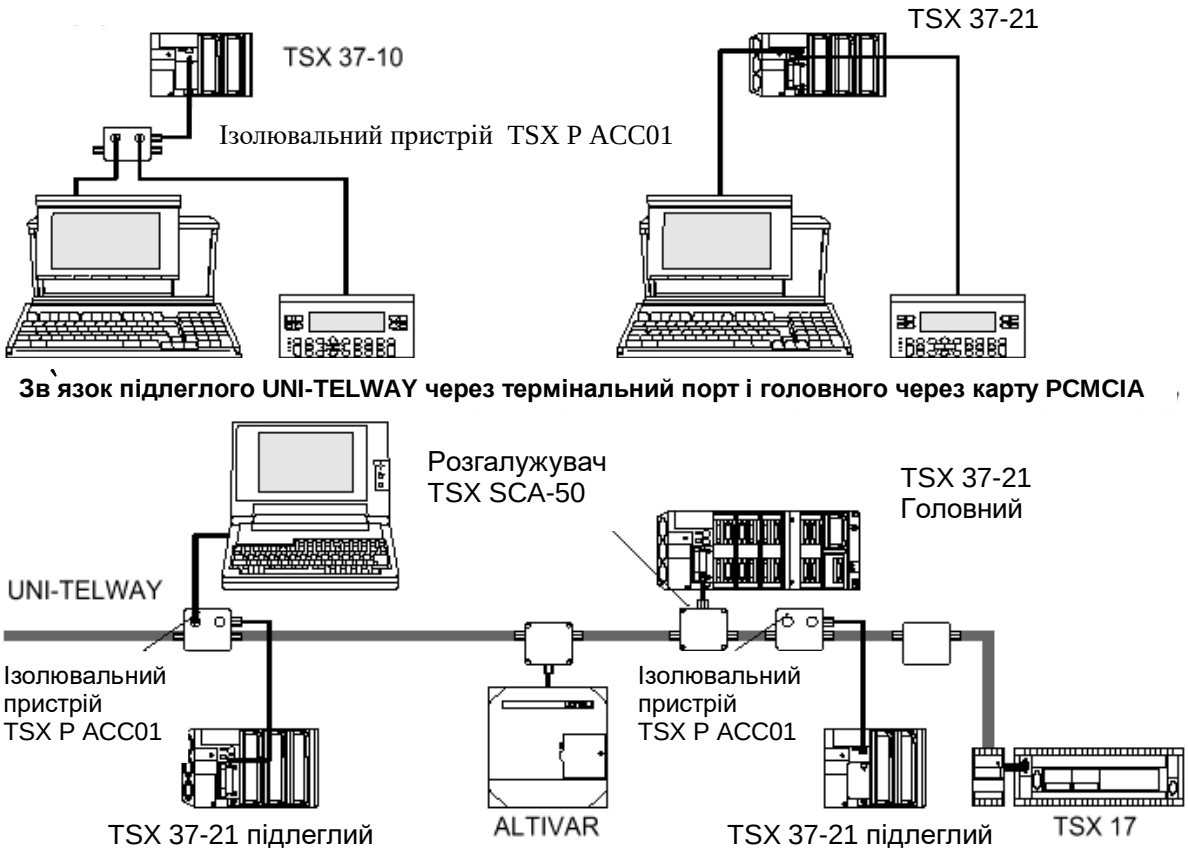


Рисунок 2.10 - Зв'язок головного UNI-TELWAY через термінальний порт і- або людино-машинний інтерфейс

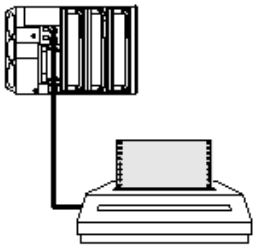
У цьому випадку термін служби виробу збільшується більше ніж на 50%.

Є три типи модулів вентилятора:

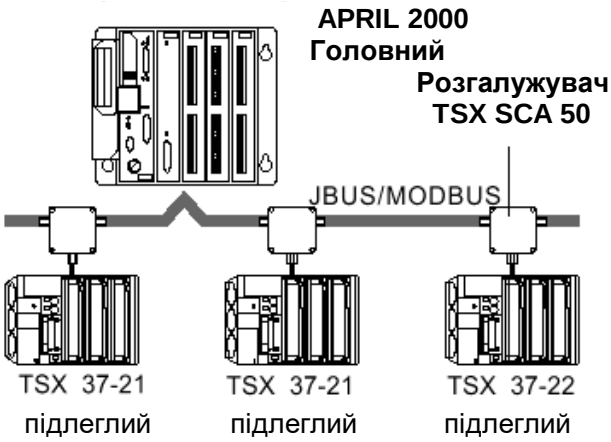
- вентилятор із напругою живлення 110 В змінного струму,
- вентилятор із напругою живлення 220 В змінного струму,
- вентилятор із напругою живлення 24 В постійного струму.

Використання примусової вентиляції не дозволяється у випадках, коли в ПЛК встановлені аналогові модулі TSX AEZ 414.

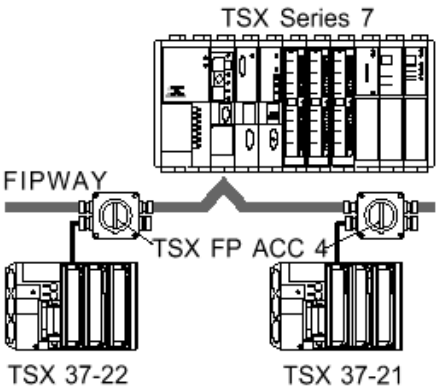
Символьний режим через термінальний порт



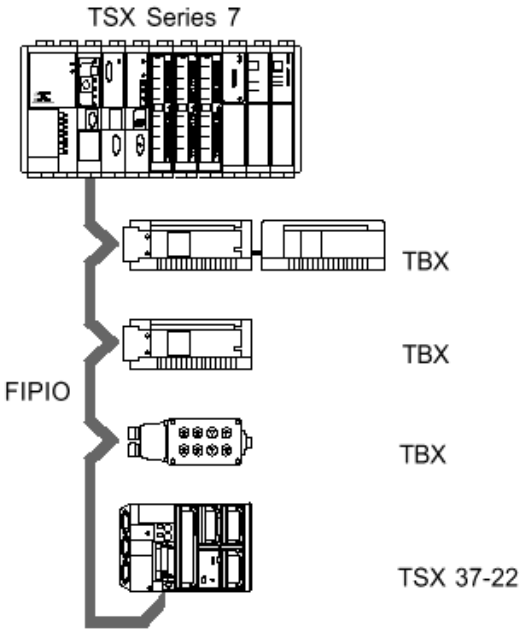
Зв`язок по JBUS/MODBUS через комунікаційну карту



Під'єднання до мережі FIPWAY через комунікаційну карту



Зв`язок по FIPIO через комунікаційну карту



Під'єднання мережі Modbus + через комунікаційну карту

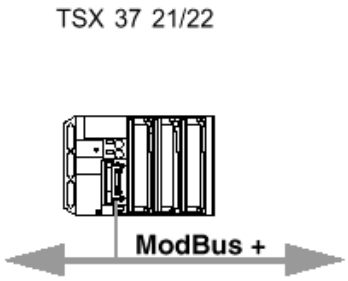


Рисунок 2.11 - Зв'язок через комунікаційну карту

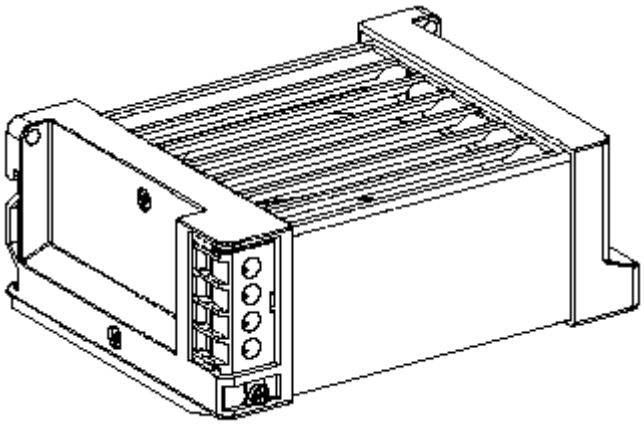


Рисунок 2.12 - Блоки примусової вентиляції

2.7 Базове виконання ПЛК TSX 37-10

2.7.1 Загальний огляд

Кожне базове шасі ПЛК 37-10 включає:

• шасі з вмонтованим джерелом живлення (24VDC або 100-240VAC), процесор, пам'ять, резервну пам'ять FLASH EPROM і два слоти для модулів;

• дискретний модуль стандартного формату на 28 або 64 I/O, встановлений у першому слоті шасі.

Таблиця 2.1 - Основні параметри дискретних модулів

Базове шасі	Напруга живлення	Вмонтований модуль I/O
TSX 3710 028AR1	100-240 VAC TSX DMZ 28AR : 16 вх. 115 VAC,	12 релейних вих.
TSX 3710 028DR1	100-240 VAC TSX DMZ 28DR : 16 вх. 24 VDC,	12 релейних вих.
TSX 3710 128DR1	24 VDC TSX DMZ 28DR : 16 вх. 24 VDC,	12 релейних вих.
TSX 3710 128DT1	24 VDC TSX DMZ 28DT : 16 вх. 24 VDC,	12 транзисторних вих.
TSX 3710 128DTK1	24 VDC TSX DMZ 28DTK :16 вх 24 VDC,	12 транзисторних вих.
TSX 3710 164DTK1	24 VDC TSX DMZ 64DTK: 32 вх 24 VDC,	32 транзисторних вих.

Використання міні-шасі розширення дає можливість додати до ПЛК два додаткових слоти. Це робить доступним 3 слоти, кожний із яких може

бути заповнений або одним модулем стандартного формату або двома напівформатними модулями.

Нижче приведені максимальні конфігурації ПЛК TSX 37-10 (максимальна кількість модулів і I/O):

Дискретні I/O	Максим. кіл-ть дискрет. I/O	у шасі + розширення + віддалені (TSX07 I/O) у шасі + розширення + віддалені (AS-i шина) у шасі у шасі + розширення віддалені (4 TSX 07) віддалені на AS-i шині (124 вх. + 124 вих.)	268 408 128 192 96 248
	Максим. кіл-ть модулів	28 або 32 дискретних I/O 64 дискретних I/O (високої щільності) віддалені I/O (для TSX 07 або AS-i шині I/O)	4 2 1
Аналогові	кіл-ть кіл-ть кіл-ть	аналогових модулів I/O (*) аналогових входів аналогових виходів	2 16 8
Лічильники	кіл-ть кіл-ть кіл-ть	500 Гц розрах. каналів на дискретних входах розрах. модулів (у шасі ПЛК) (*) 40 кГц або 500 Гц розрахун. каналів	2 2 4

(*) Лічильні модулі можуть бути встановлені тільки в шасі ПЛК. У ПЛК TSX 37-10 можуть бути встановлені 2 аналогові модулі і 2 лічильні модулі.

Термінальний порт RS 485 із 8-ми штирковим роз'ємом стандарту DIN дозволяє:

• під’єднати до ПЛК програмний термінал типу FTX або PC - сумісну ПЕОМ або принтер;

• під’єднати ПЛК до шини UNI-TELWAY. При під’єднанні до UNI-TELWAY за замовчуванням ПЛК конфігурується як головний із швидкістю передачі 9600 бод. ПЛК можна сконфігурувати також для роботи як ведений UNI-TELWAY або в символьному ASCII режимі.

Ізолювальний пристрій дозволяє одночасно підключити до шини UNI-TELWAY ПЛК і програмувальний термінал. Він також повинен використовуватися, коли відстань між пристроями по шині UNI-TELWAY більша 10 метрів.

2.7.2 Дисплейний блок

Дисплейний блок 1 (рисунок 2.13) призначений для відображення інформації, необхідної для діагностики і підтримки ПЛК і його модулів. Для цього він має:

• 8 індикаторних ламп стану, що відображають інформацію про режими роботи і функціонування ПЛК (індикаторні лампи RUN, TER, I/O, ERR, BAT), а також про режими роботи дисплея (R I/O, WRD і DIAG);

• блок із 96 індикаторних ламп, що можуть відображати:

- у режимі відображення локальних I/O (горять індикаторні лампи BASE або EXT): стан усіх входів-виходів базового шасі ПЛК і міні-шасі розширення,

- у режимі відображення віддалених I/O (горять індикаторні лампи R I/O): стан дискретних I/O кожного веденого на AS-i шині,

- у режимі діагностики (горить індикаторна лампа DIAG):

а) для локальних I/O: помилка модуля (всі індикаторні лампи, пов'язані з цим модулем, повільно мигають) або помилка каналу (індикаторна лампа, пов'язана з цим каналом, мигає швидко),

б) для віддалених I/O на AS-i шині: стан кожного веденого I/O (несправний ведений I/O мигає).

- у режимі відображення об'єктів (горить індикатор WRD): значення максимум 16 слів %MWi, %SWi або %KWi (ці слова відображаються в двійковому або шістнадцятковому форматі); стан групи з 64 бітів %Mi, %Si or %Ki; стан бітів входів і виходів модулів TSX 07, використовуваних як дискретні віддалені I/O.

• кнопки, що дозволяють змінювати режими роботи дисплейного блока і дозволяти відображення інформації.

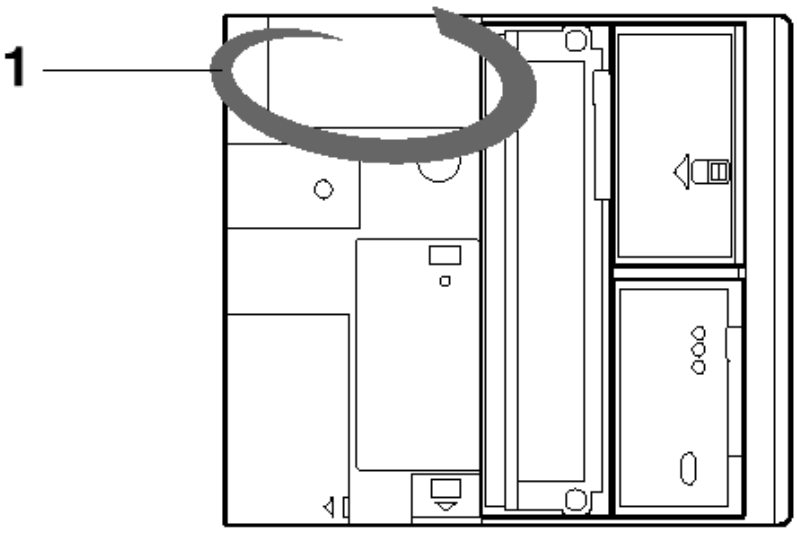


Рисунок 2.13 - Дисплейний блок

2.7.3 Загальний вигляд

Загальний вигляд TSX 37-10 представлено на рисунку 2.14.

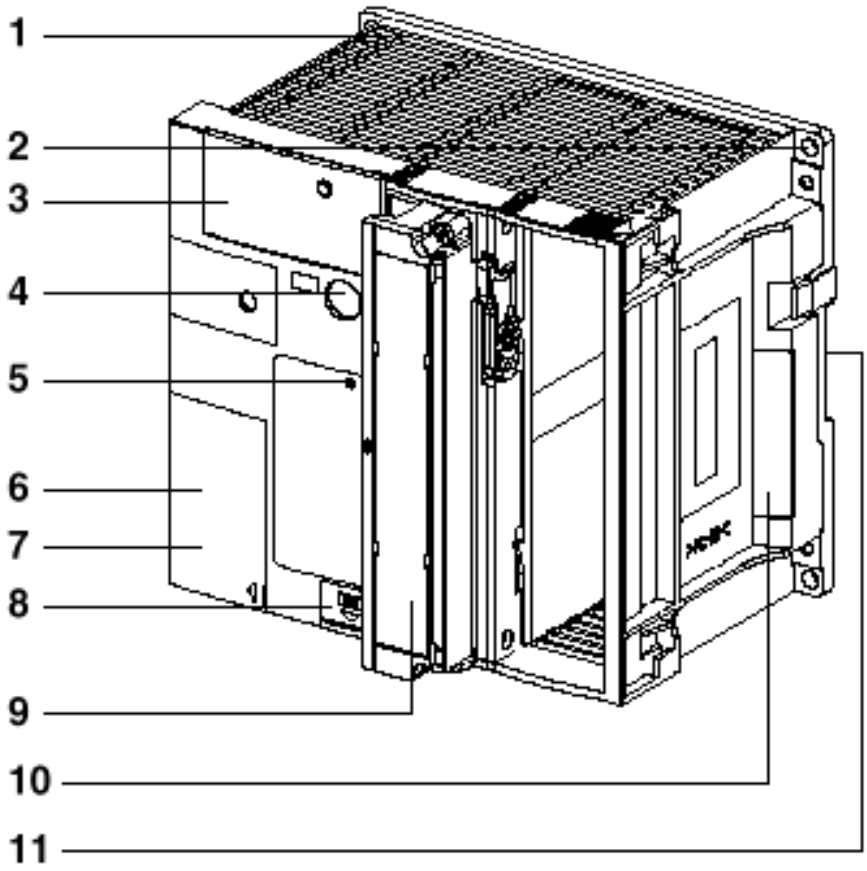


Рисунок 2.14 - Загальний вигляд TSX 37-10

TSX 37-10 складається із:

1 - шасі з двома слотами для модулів, вмонтоване джерело живлення, процесор з його пам'яттю.

2 - монтажні отвори ПЛК.

3 - дисплейний блок.

4 - термінальний порт (TER).

5 - кнопка RESET (скидання).

6 - кришка для доступу до контактів джерела живлення.

7 - ярлик, що заповнюється при заміні батареї.

8 - кришка для доступу до додаткової батареї і перемикача захисту операційної системи.

9 - модуль із 28 або 64 входами-виходами, розміщений у першому слоті.

10 - кришка доступу до з'єднувача для включення міні-шасі.

11 - пристрій для монтажу на DIN-рейці.

Для забезпечення ступеня захисту IP20, вільні слоти закриваються спеціальними кришками TSX RKA 01.

2.8 Базове виконання TSX 37-21 і TSX 37-22

2.8.1 Загальний огляд

ПЛК TSX 37-21 або TSX 37-22 містять шасі з вмонтованим джерелом живлення 24 В постійного струму (TSX 37 21 101 і TSX 37 22 101) або 100-240 В змінного струму (TSX 3721001 і TSX 37 22 001), процесор, пам'ять, систему резервного копіювання і 3 слоти для модулів.

Використання міні-шасі розширення TSX RKZ 02 дає можливість додати до ПЛК 2 додаткових слоти і, таким чином, одержати 5 слотів для встановлення модулів I/O. У кожному із 5 слотів може бути встановлений або один модуль стандартного формату або два напівформатних модулі (за винятком першого слоту, у який можуть встановлюватися тільки модулі стандартного формату).

Нижче приведені максимальні конфігурації ПЛК TSX 37-21 і TSX 37-22 (максимальна кількість модулів I/O):

ПЛК	TSX		37-21	37-22
Дискретні	**Максимальна кількість входів-виходів**	базові+розширення+віддалені (TSX 07)	332	332
		базові+розширення+віддалені (AS-i шина)	472	472
		у базовому шасі ПЛК	192	192
		у базовому + міні-шасі розширення	256	256
		віддалені (4 TSX 07)	96	96
		віддалені на AS-i шині (124 вх. + 124 вих.)	248	248
	Максимальна кількість модулів	28 або 32 дискретних входів-виходів	5	5
		64 дискретних вх.-вих. (висока щільність)	3	3
		для віддалених вх-вих(TSX 07 або AS-i шина)	1	1
Аналогові	Максим. кіл-ть модулів аналогових вх-вих.		4	4
	Максим. кіл-ть аналогових входів у базовому шасі		32	32
	Максим. кіл-ть аналогових виходів у базовому шасі		16	16
	Максим. кіл-ть інтегрованих аналогових входів		_	8
	Максим. кіл-ть інтегрованих аналогових виходів		_	1
Лічильні	Максим. кіл-ть 500 Гц лічильних каналів на дискр. входах		2	2
	Максим. кіл-ть лічильних модулів (у ПЛК) (1)		4	4
	Максим. кіл-ть 40 кГц і/або 500 кГц лічильних каналів		7	7
	Максим. кіл-ть інтегрованих лічильних каналів (10 кГц)		_	2
Комунікації (2)	Кіл-ть комунікаційних карт (виділений слот)		1	1

(1) Лічильні модулі можуть бути встановлені тільки в базове шасі ПЛК. У TSX 37-21/22 можуть бути встановлені 4 аналогових і 4 рахункових модулі.

(2) Комунікаційні карти формату PCMCIA (FIPWAY, Agent FIPIO, Modbus+).

Два термінальних порти RS 485 із 8-штирковими роз'ємами стандарту DIN дозволяють підключати до ПЛК:

• **TER**: програмний термінал FTX або PC-сумісну ПЕОМ, а також підключати ПЛК до шини UNI-TELWAY через ізолювальний пристрій TSX P ACC 01;

• **AUX**: людино-машинний інтерфейс або принтер.

За замовчуванням, для роботи в UNI-TELWAY обидва порти сконфігуровані в режимі ведучого на швидкості 9600 бод. Вони також можуть бути сконфігуровані для роботи в UNI-TELWAY у режимі веденого або в символьном режимі ASCII.

Дисплейний блок відображає всю необхідну інформацію для діагностики і підтримки ПЛК (базового шасі і міні-шасі розширення) та його модулів.

Два слота формату PCMCIA призначені для карт розширення пам'яті і комунікаційної карти.

ПЛК TSX 37-22 також має 3 з'єднувачі для під’єднання інтегрованих аналогових I/O і лічильників.

2.8.2 Загальний вигляд

Загальний вигляд TSX 37-21 та TSX 37-22 поданий на рисунках 2.15-2.16.

TSX 37-21 і TSX 37-22 складаються з:

1 - шасі з трьома слотами для модулів, вмонтоване джерело живлення, процесор з його пам'яттю.

2 - монтажні отвори ПЛК.

3 - дисплейний блок.

4 - термінальний порт TER.

5 - порт для під’єднання людино-машинного інтерфейсу.

6 - слот для карти розширення пам'яті. Якщо карта не встановлена, цей слот закривається кришкою, яка не повинна зніматися. Її зняття викликає зупинку ПЛК.

7 - кришка для доступу до контактів джерела живлення.

8 - ярлик, що заповнюється при зміні батарейки.

9 - контакти джерела живлення.

10 - слот для комунікаційної карти.

11 - кришка для доступу до додаткової батареї і перемикача захисту операційної системи.

12 - з'єднувач для під’єднання міні-шасі, звичайно закритий кришкою, що знімається.

13 - пристрій для монтажу на DIN рейці.

14 - з'єднувач для під’єднання вмонтованих аналогових і лічильних функцій.

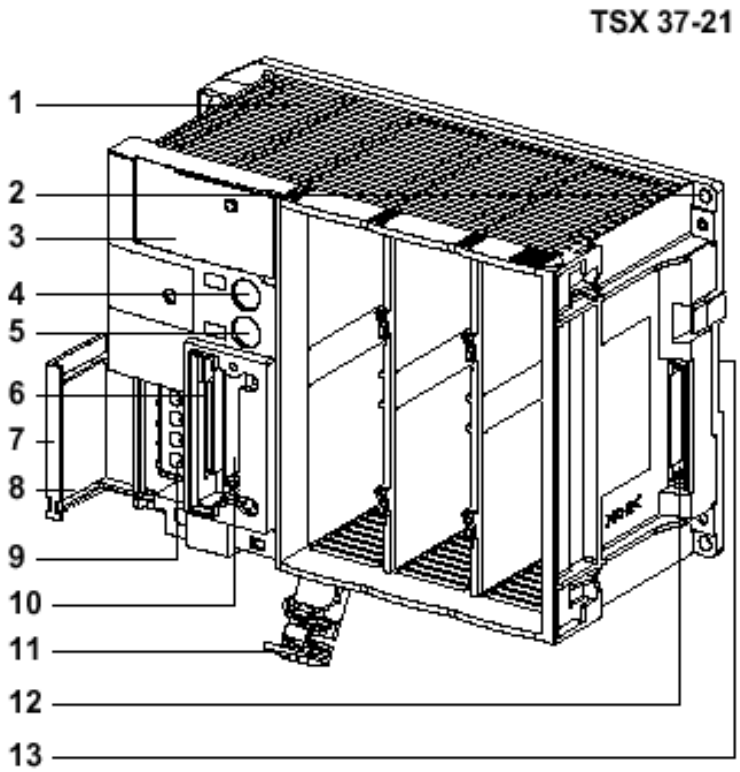


Рисунок 2.15 - Загальний вигляд TSX 37-21

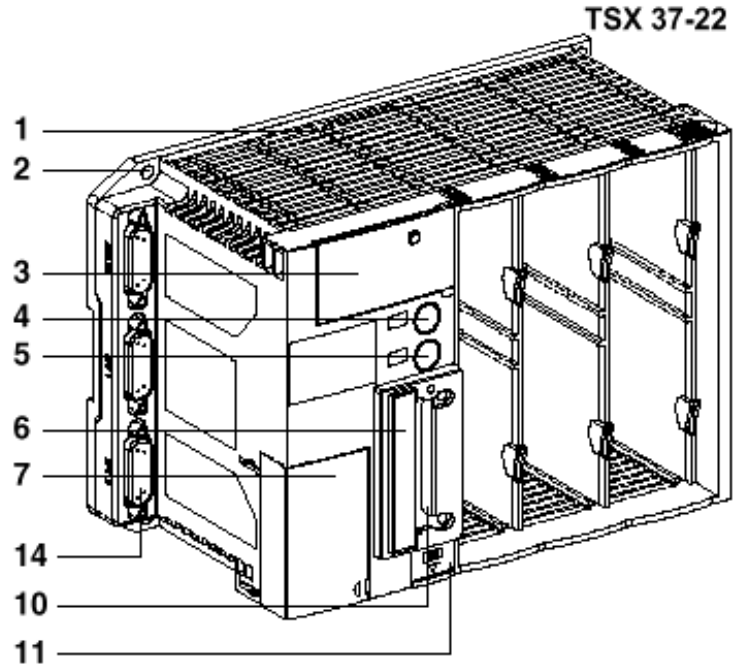


Рисунок 2.16 - Загальний вигляд TSX 37-22

2.9 Основні технічні параметри TSX Micro

Тип ПЛК		TSX37-10	TSX37-21	TSX37-22
Функції	К-ть дискр. вх.-вих. ПЛК + віддалені TSX 07 ПЛК+ віддалені AS-і шини	 268 408	 332 472	 332 472
	К-ть вмонтованих UNI-TELWAY з'єднань	1	1	1
	Комунікаційні модулі	0	1	1
	Астрономічний годинник	немає	є	є
	Вмонтовані аналог. вх-вих	немає	немає	є
	Вмонтовані 500 Гц лічильники 10 кГц	є немає	є немає	є є
Пам'ять	Внутр.енергонезалежн.RAM • програма • дані • константи	14Кслів 4.7Кінст. 1 Кслів 128 слів	20 Кслів 7.9Кінстр. 2 Кслів 128 слів	20 Кслів 7.9Кінст. 2 Кслів 128слів
	Вмонтована Flash Eprom	16 Кслів	16 Кслів	16 Кслів
	Карта PCMCIA 32 K16 • програма • дані (у внутр. RAM) • константи	_	32 Кслів 18.5Кін. 17.5Кслів 128 слів	32 Кслів 18.5Кін 17.5Кслів 128 слів
	Карта PCMCIA 64 K16 • програма • дані (у внутр. RAM) • константи	_	64 Кслів 40 Кінс. 17.5Кслів 128 слів	64 Кслів 40 Кінс 17.5Кслів 128 слів
Час вик. Кінс.	RAM PCMCIA	0.3 мс _	0.15 мс 0.225 мс	0.15 мс 0.225 мс
Системні витрати часу		1.9 мс	1.6 мс	2.3 мс
Структура програми	Основна задача Швидка задача Обробка подій	1 1 1..8	1 1 1..16	1 1 1..16
Функціональні блоки	Таймери Лічильники	64 32	64 32	64 32

2.10 Міні-шасі розширення

Міні-шасі розширення TSX RKZ 02 дозволяє додати до ПЛК два слоти, у кожний із яких можна встановити або один стандартний, або два напівформатні модулі. Воно складається із (рисунок 2.17):

1- шасі розширення з двома слотами.

2 - монтажні отвори для міні-шасі.

3 - гвинти для кріплення міні-шасі до базового.

4 - індикаторна лампа, що показує наявність додаткової напруги 24 В (для релейних або аналогових модулів).

5 - контакти джерела живлення, закриті кришкою, яка знімається.

6 - клема заземлення.

7 - роз'єм для під’єднання до базового ПЛК (об'єднуюча шина і продовження заземлення.

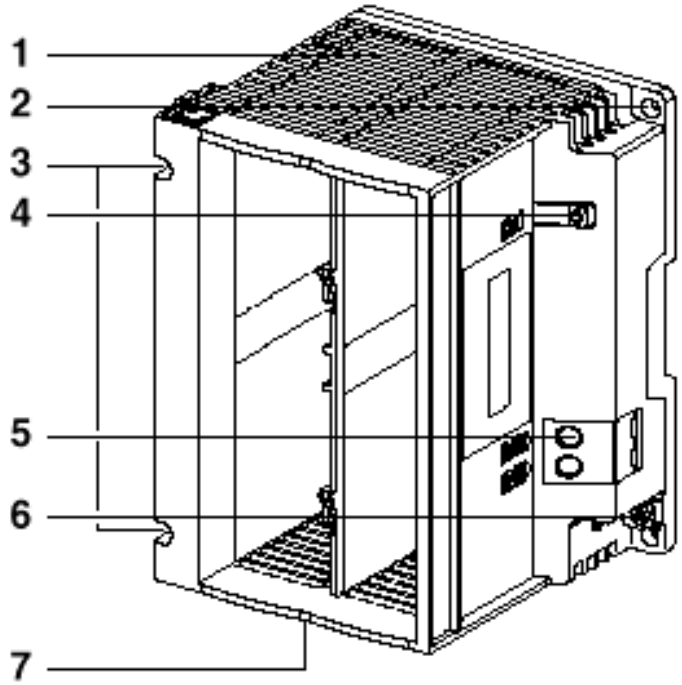


Рисунок 2.17 - Міні-шасі розширення TSX RKZ 02

Коли ПЛК TSX 37-10/21/22 живиться від джерела живлення змінного струму, останній не забезпечує напругу живлення 24 В постійного струму для міні-шасі розширення. У цьому випадку, якщо в шасі розширення є релейні або аналогові модулі, додаткове джерело живлення 24 В постійного струму повинно бути підімкнене до відповідних контактів міні-шасі розширення (рисунок 2.18).

Джерело живлення 24 В базового шасі забезпечує живлення входів-виходів міні-шасі, які його потребують, забезпечуючи споживання <=400мА. Якщо планується більше споживання, то необхідно використовувати додаткове джерело живлення.

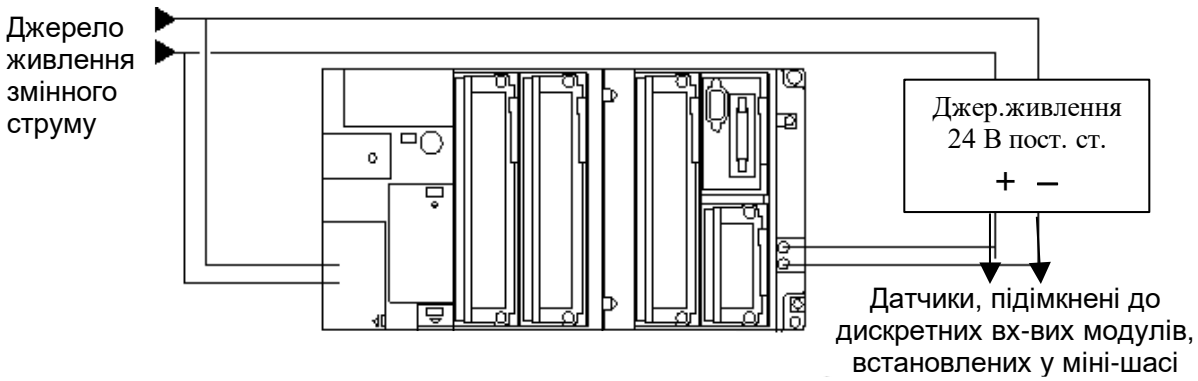


Рисунок 2.18 - Підключення джерела живлення до міні-шасі розширення

Список питань для самоконтролю

1. Наведіть основні характеристики TSX Micro.
2. Дайте характеристику дискретним модулям входів-виходів.
3. Дайте характеристику аналоговим модулям входів-виходів.
4. Дайте характеристику лічильним каналам TSX Micro.
5. Дайте характеристику комунікаційним можливостям TSX Micro.
6. Наведіть основні параметри і будову базової моделі ПЛК TSX 31-10.
7. Наведіть основні параметри і будову базових моделей ПЛК TSX 31-21 і TSX 31-22.
8. Як використовується міні-шасі розширення?

3 СТРУКТУРА ТА АЛГОРИТМИ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПЛК MODICON TSX MICRO

3.1 Адресація каналів

Адресація каналів (рисунок 3.1) географічна, тобто залежить від фізичного розташування модуля в ПЛК або шасі розширення.

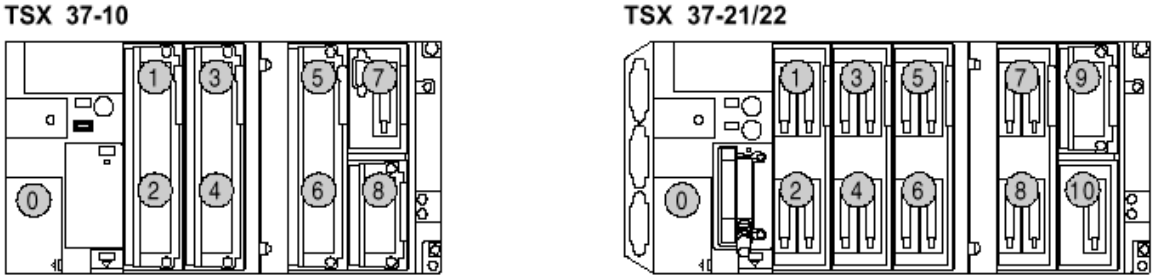


Рисунок 3.1 - Адресація каналів TSX Micro

Оскільки базовим при адресації є модуль половинного формату, модулі стандартного формату адресуються як 2 об'єднаних модулі половинного формату. Далі в цьому розділі термін ПОЗИЦІЯ (модуля) означає або модуль половинного формату або верхню чи нижню частину модуля стандартного формату.

Синтаксис описання дискретних I/O наведений на рисунку 3.2.

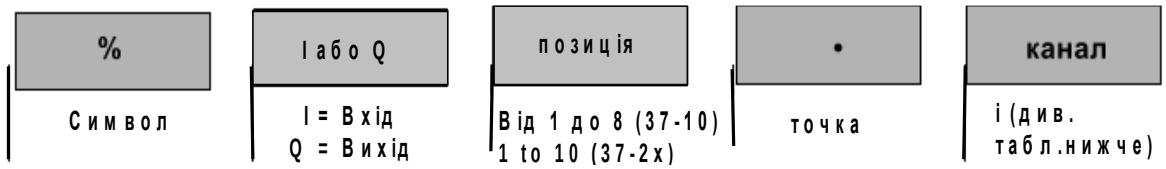


Рисунок 3.2 - Синтаксис описання дискретних I/O

Приклади:

%I1.5 означає: вхід 5 модуля, розміщеного в позиції 1 (рисунок 3.3, а),

%Q8.3 означає: вихід 3 модуля, розміщений у позиції 8 (рисунок 3.3, б).

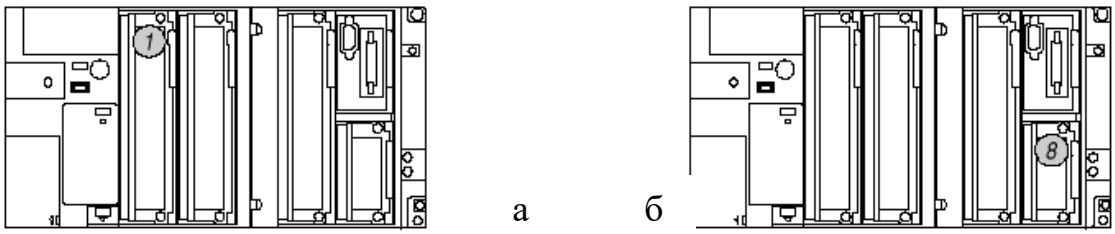


Рисунок 3.3 - Ілюстрація до прикладу

Тип модуля	64 вх/вих	32 вх	32 вих	28 вх/вих	12 вх	8 вих	4 вих
Номер каналу : i	0..31 0..31	0..15 0..15	0..15 0..15	0..15 0..11	0..11	0..7	0..3
Адреси каналів	%Ix.0 до %Ix.31	%Ix.0 до %Ix.15	%Qx.0 до %Qx.15	%Ix.0 до %Ix.15	%Ix.0 до %Ix.11	%Qx.0 до %Qx.7	%Qx. 0 до %Qx.3
	%Q(x+1).0 до %Q(x+1).31	%I(x+1).0 до %I(x+1).15	%Q(x+1).0 до %Q(x+1).15	%Q(x+1).0 до %Q(x+1).11			

Канали модуля 0: 0 - термінальний комунікаційний порт, 1 - комунікаційний порт PCMCIA (TSX 37-21/22), від 2 до 9 - вмонтовані аналогові входи (TSX 37-22), 10 - вмонтований аналоговий вихід (TSX 37-22), 11 і 12 - вмонтовані лічильні канали.

3.2 Адресація слів та бітів

Синтаксис адресації слова поданий на рисунку 3.4.

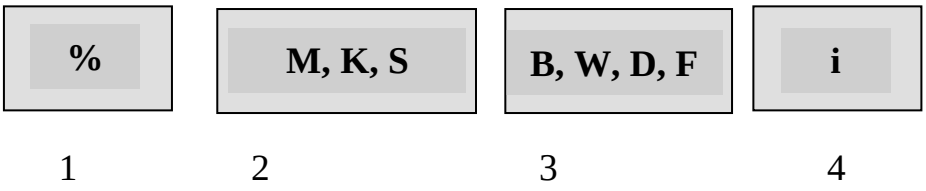


Рисунок 3.4 - Адресація слова

1 – символ.

2 – тип об'єкта:

M – внутрішнє слово, в якому запам'ятовуються числа, що обчислюються програмою.

K – константи, символьні повідомлення. Вони записуються і модифікуються в процесі написання і відлагодження програми.

S – системні біти і слова. Ці слова мають кілька функцій: відображення інформації про стан виконання програми; відображення системних параметрів (час, дата і т.д.).

3 – формат:

B – байт. Даний формат використовується для операцій з символьними рядками.

W – слово одинарної довжини 16 біт і змінюється в діапазоні –32 768...+32 767.

D – подвійне слово 32 біта і змінюється в діапазоні

–2 147 483 648...+2 147 487 647.

Дане слово займає в пам'яті два слова одинарної довжини.

F – число з плаваючою точкою 32 біта.

4 – Номер.

Байти, слова, подвійні слова і числа з плаваючою точкою записуються у пам'яті в одній зоні (рисунок 3.5).

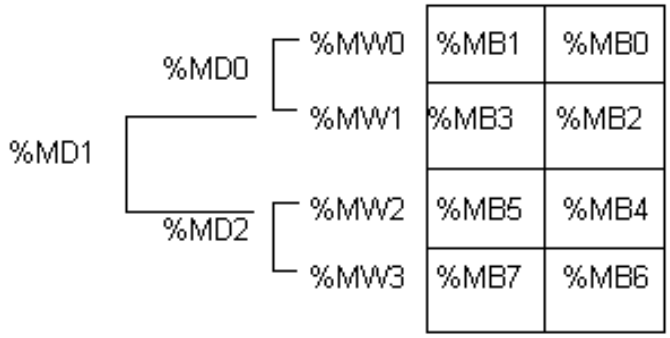


Рисунок 3.5 - Розподіл пам'яті

Слово подвійної довжини %MDi або %MFi відповідає словам %MWi (молодший байт) і %MWi+1 (старший байт).

Синтаксис адресації бітів у слові поданий на рисунку 3.6.

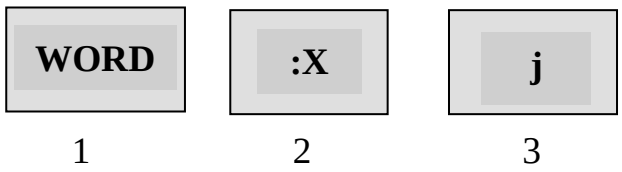


Рисунок 3.6 - Адресація бітів у слові

1 – адресоване слово.

2 – символ.

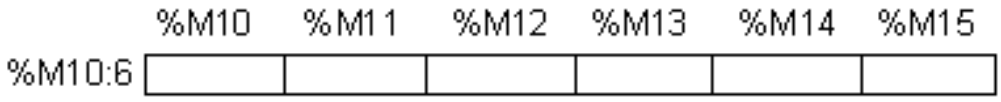
3 – позиція, j = 0..15.

Приклад: %MW10:X4 – біт 4 слова %MW10.

3.3 Структуровані об'єкти

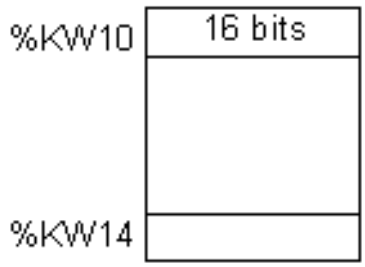
Таблиця бітів – послідовність суміжних бітів певного типу і визначеної довжини L.

Приклад: таблиця, яка вміщує 6 бітів - **%M10:6**



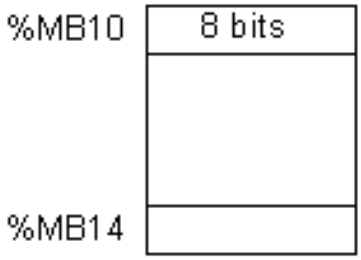
Таблиця слів – послідовність суміжних слів певного типу і визначеної довжини L.

Приклад: таблиця констант типу слова, яка вміщує 5 слів – **%KW10:5**



Таблиця символів – послідовність суміжних байтів певного типу і визначеної довжини L.

Приклад: таблиця символів, яка вміщує 5 символів – **%MB10:5**



3.4 Індексовані об′єкти

Використовується два типа адресації – пряма та індексована.

Приклад прямої адресації: %MW26 (внутрішнє слово за адресою 26).

Приклад індексованої адресації: %MW108[%MW2] (пряма адреса 108 + вміст слова %MW2). Якщо %MW2=12, запис %MW108[%MW2] означає %MW[108+12] $\rightarrow$ %MW120.

В індексованій адресації вміст індекса додається до прямої адреси об′єкта. Номер індекса не обмежений.

Приклад: %MD6[%MW100].

Якщо використовується адресація слів подвійної довжини, то вміст індекса множиться на 2:

(%MW100=10) $\rightarrow$ 6 + 2 · 10 = %MW26.

3.5 Цикл ПЛК

3.5.1 Циклічне виконання (тільки MAST-задача)

Цей тип роботи відповідає звичайному виконанню циклу ПЛК (встановлюється за замовчуванням). Для цього типу роботи характерно

послідовне виконання циклів основної задачі (MAST) один за другим (рисунок 3.7).

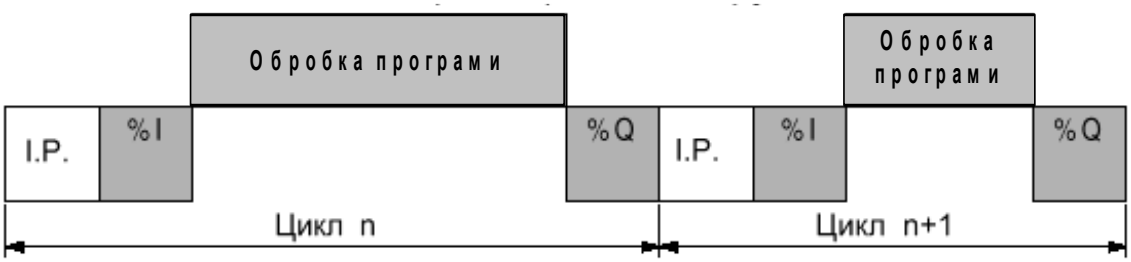


Рисунок 3.7 - Циклічне виконання програми

I.P. - Внутрішня обробка: система неявно керує ПЛК (системними бітами і словами, обновляючи стан астрономічного годинника, обновляючи стан індикаторних ламп, детектуючи зміни RUN/STOP, і т.п.) і оброблюючи запити від програмувального термінала або комунікаційної системи;

%I - Зчитування входів: записує інформацію про стан фізичних входів у пам'ять;

Обробка програми - виконує прикладну програму, написану користувачем;

%Q - обновлення виходів: записує вихідні біти і слова, пов'язані з дискретними та спеціальними для програми модулями, згідно зі станом, розрахованим прикладною програмою.

Робочий цикл

ПЛК у стані RUN: процесор виконує, один за одним, внутрішню обробку, зчитування входів, обробку прикладної програми і відновлення виходів.

ПЛК у стані STOP: процесор виконує тільки внутрішню обробку і зчитування входів. Відновлення виходів виконується відповідно до конфігурації режиму нейтралізації несправності кожного дискретного або аналогового модуля.

• нейтралізації в 0: фізичні виходи модуля примусово встановлюються в 0 (відображення в пам'ять не змінюється),

• підтримуваний стан: фізичні виходи модуля підтримуються в їх останньому стані.

Перевищення часу виконання

Тривалість виконання прикладної програми відслідковується ПЛК (сторожовий таймер) і не повинна перевищити значення, занесеного при конфігурації в системне слово %SW11. У випадку перевищення системний біт %S11 встановлюється в 1 і програма оголошується помилковою, що викликає негайну зупинку ПЛК (індикаторні лампи ERR і RUN мигають).

Рисунок 3.8 - Алгоритм роботи ПЛК при циклічному виконанні програми

3.5.2 Періодичне виконання

У цьому робочому режимі зчитування входів, обробка прикладної програми і відновлення виходів виконуються періодично відповідно до часу, визначеному при конфігурації (від 1 до 255 мс) у системному слові %SW0.

Проте, рекомендується використовувати такі мінімальні часи: 3 мс для MAST і 2 мс для FAST задач.

На початку циклу ПЛК таймер ініціалізується значенням, заданим у конфігурації, починає відлік на віднімання. Цикл ПЛК повинен закінчитися перед цим закінченням відліку, після якого починається новий цикл.

I.P. - Внутрішня обробка: система неявно управляє ПЛК (системними бітами і словами, обновляючи стан астрономічного годинника, обновляючи стан індикаторних ламп, детектуючи зміни RUN/STOP, і т.п.) і обробляючи запити від програмувального термінала або комунікаційної системи.

%I - Зчитування входів: записує інформацію про стан фізичних входів в пам'ять.

Обробка програми - виконує прикладну програму, написану користувачем.

% Q - поновлення виходів: записує вихідні біти і слова, пов'язані з дискретними і спеціальними для програми модулями, згідно зі станом, розрахованим прикладною програмою.

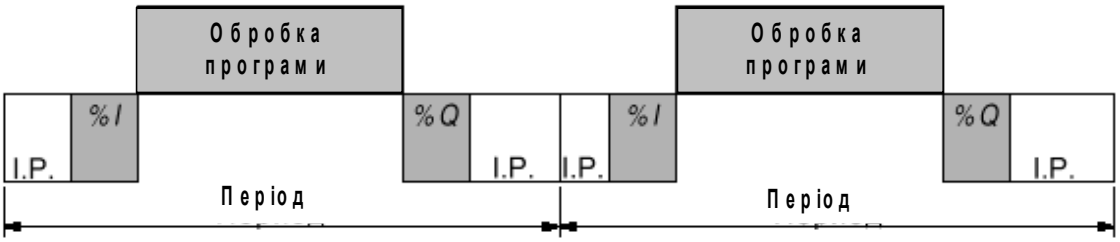


Рисунок 3.9 - Періодичне виконання програми

Цикл роботи

ПЛК у стані RUN: процесор виконує, один за одним, внутрішню обробку, зчитування входів, обробку прикладної програми і відновлення виходів. Якщо період ще не закінчений, процесор заповнює свій операційний цикл системними або фоновими задачами до кінця періоду.

Якщо час виконання перевищує час, визначений у конфігурації, ПЛК сигналізує про перевищення установкою системного біта %S19 в 1; обробка продовжується і закінчується повністю (якщо не було перевищення часу, що відслідковується сторожовим таймером).

Такий цикл починається після неявного запису виходів поточного циклу.

ПЛК у стані STOP: процесор виконує тільки внутрішню обробку і читання входів. Відновлення виходів виконується відповідно до конфігурації режиму нейтралізації несправності кожного дискретного або аналогового модуля.

• нейтралізації в 0: в фізичні виходи модуля примусово встановлюються 0 (відображення в пам'ять не змінюється),

• підтримуваний стан: фізичні виходи модуля підтримуються в їх останньому стані.

Перевищення часу виконання

Тривалість виконання прикладної програми (при циклічному або періодичному виконанні) відслідковується ПЛК (сторожовий таймер) і не повинна перевищити значення, занесеного при конфігурації в системне слово %SW11. У випадку перевищення системний біт %S11 встановлюється в 1, і додаток оголошується помилковим, що викликає негайну зупинку ПЛК (індикаторні лампи ERR і RUN мигають).

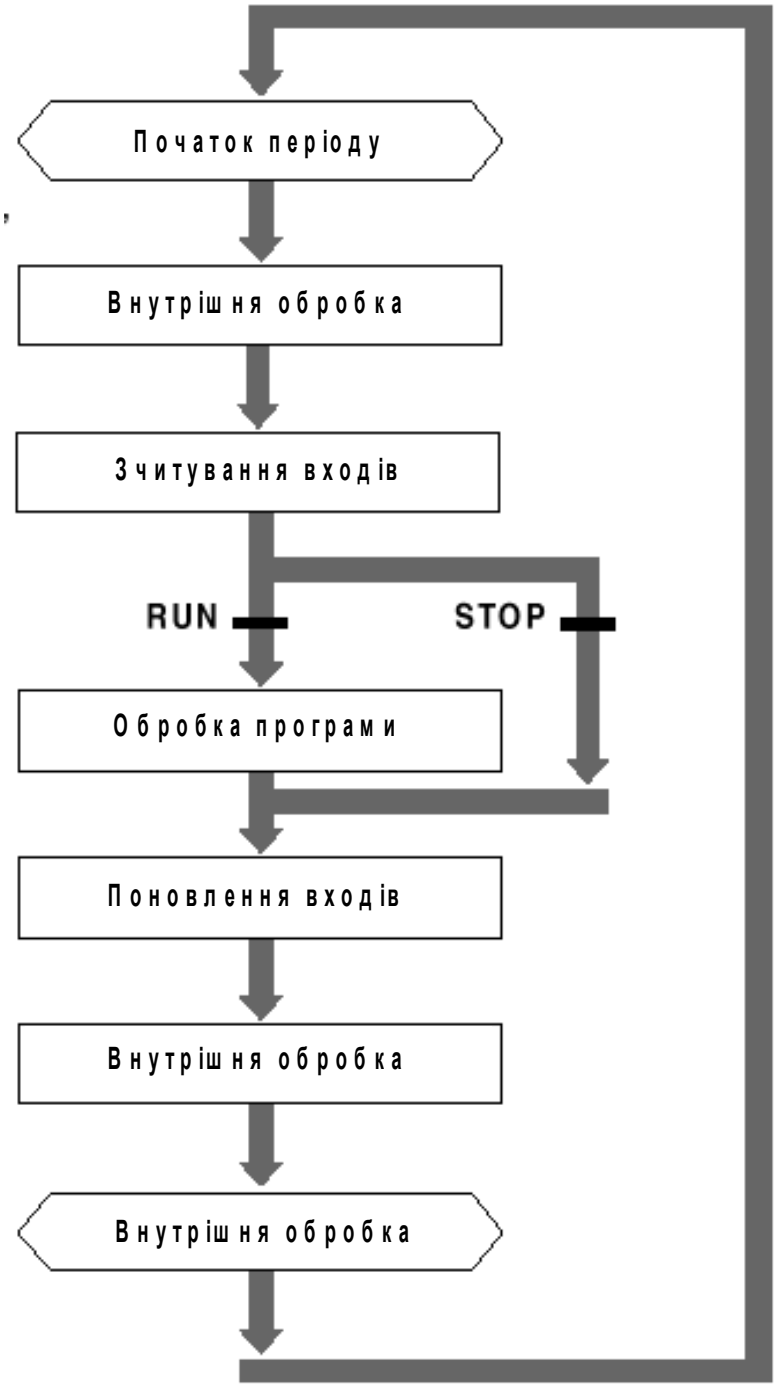


Рисунок 3.10 - Алгоритм роботи ПЛК при періодичному виконанні програми

3.6 Структура програми

Структура програми ПЛК TSX 37-10, TSX 37-21 або TSX 37-22 ПЛК може бути одно- або багатозадачною. В однозадачній структурі є тільки основна задача (MAST), використовувана в режимі циклічного або

періодичного виконання. У багатозадачній структурі є 2 керуючі задачі (MAST і FAST) і декілька задач обробки подій.

Задачі виконуються відповідно до пріоритету кожної з них. На початку виконання задачі (настання події або початок циклу) зупиняється виконання менш важливих задач. Після виконання задачі з більш високим пріоритетом продовжується виконання перерваної задачі.

Зразкова структура задачі така:

• основна задача MAST, низький пріоритет, є завжди, виконується циклічно або періодично,

• швидка задача FAST, середній пріоритет, є необов'язковою, виконується завжди періодично,

• від 1 до 8 задач обробки подій EVTi (TSX 37-10) або від 1 до 16 задач обробки подій EVTi (TSX 37-21/22), високий пріоритет, викликаються системою при настанні подій. Ці задачі необов'язкові і корисні для програм, що потребують швидкої реакції.

У ПЛК TSX 37-21/22 задача обробки події EVT0 має більш високий пріоритет, ніж задача EVT1- EVT15.

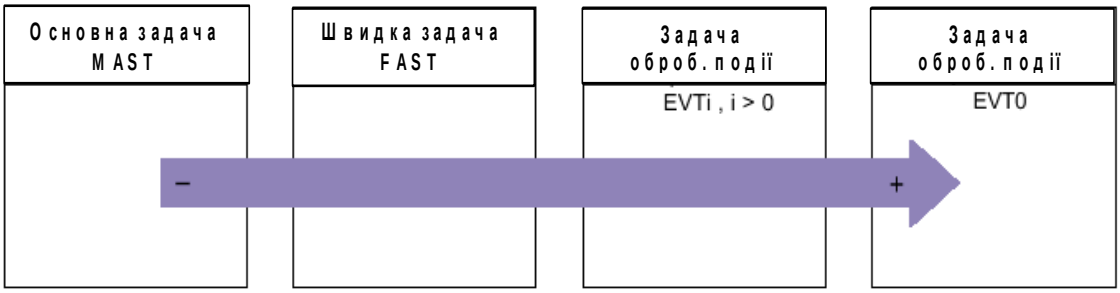


Рисунок 3.11 - Структура задач ПЛК TSX Micro

Приклад багатозадачної обробки (рисунок 3.12):

• циклічна основна задача (MAST);

• швидка задача з періодом 20 мс (FAST);

• задача обробки події (Event).

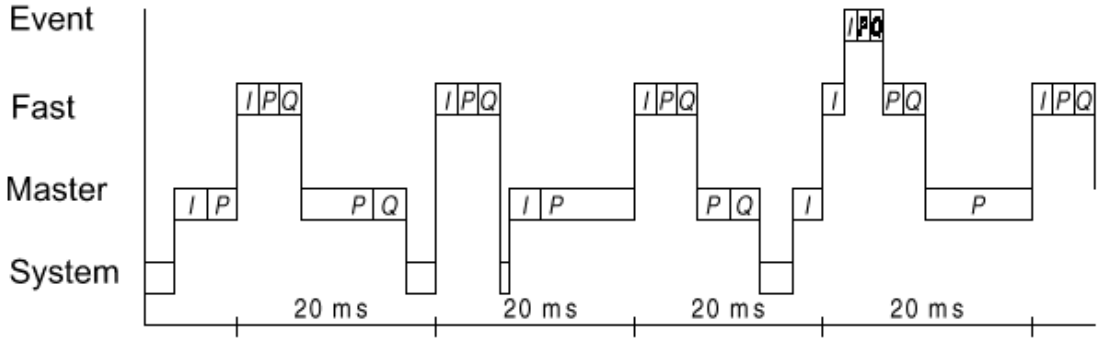


Рисунок 3.12 - Багатозадачна обробка

3.6.1 Керуючі задачі

Основна задача MAST. Ця задача має самий найнижчий пріоритет і містить основну частину програми. Вона керується системним словом %SW0 (із переналагоджуваною конфігурацією: рекомендовано мінімум 3мс), що дозволяє функціонування в циклічному (за замовчуванням) або періодичному режимах. Задача MAST організована відповідно до моделі, описаної в попередньому параграфі: безпосереднє читання входів, виконання прикладної програми і безпосереднього запису виходів. Будь-який режим періодичний або циклічний, контролюється сторожовим таймером, що дозволяє виявити ненормальну тривалість задачі. У випадку його перевищення системний біт %S11 встановлюється в 1, і програма оголошується помилковою.

FAST задача. Ця задача має більш високий пріоритет, ніж задача MAST і є періодичною, щоб дозволити виконуватися задачі з більш низьким пріоритетом. Задача FAST керується системним словом %SW1 (із переналагоджуваною конфігурацією: рекомендовано мінімум 2 мс), у якому можна визначити період. Період може бути довшим, ніж чим у задачі MAST, щоб адаптуватися до повільної періодичної обробки. Проте час виконання програми повинен бути коротким, щоб не перешкоджати основній задачі.

Примітка:

Коли швидка задача не програмується, вона не існує в ПЛК, і системні біти і слова, пов'язані з нею, не значущі.

Перевищення періоду.

При періодичному виконанні (задача MAST і FAST), якщо операційний час перевищує установлене значення, системний біт %S19 у ПЛК встановлюється в 1, що є сигналом перевищення часу періоду. Обробка продовжується і проводиться цілком (час не повинен перевищити уставку сторожового таймера). Такий цикл почнеться після неявного запису виходів поточного циклу.

Присвоювання каналу або групі каналів задачі визначається в екрані конфігурації відповідного модуля. За замовчуванням всі канали присвоюються задачі MAST.

Каналам дискретних модулів можуть бути присвоєні задачі MAST або FAST групами по 8 послідовних каналів (від 0 до 7, від 8 до 15, і т.д).

Наприклад, можливо призначити 28 каналів модуля I/O в такий спосіб:

- входи від 0 до 7 присвоєні задачі MAST;
- входи від 8 до 15 присвоєні задачі FAST;
- виходи від 0 до 7 присвоєні задачі MAST;
- виходи від 8 до 11 присвоєні задачі FAST.

Кожен канал лічильного модуля може бути присвоєний задачі FAST або MAST.

Наприклад, для двоканального лічильного модуля можливо присвоїти канал 0 задачі MAST, а канал 1 - задачі FAST.

Канали аналогових вхідних модулів повинні бути присвоєні задачі MAST.

Деякі аналогові канали вводу можуть бути присвоєні задачі MAST або FAST (стосується каналів, інтегрованих в основний модуль і каналів модулів TSX AEZ 801/802). Проте, можливо присвоїти аналогові вихідні канали задачі MAST або FAST групами по 2 канали. Наприклад, для модуля з 4-ма аналоговими виходами можливо присвоїти канали 0 і 1 задачі MAST, а канали 2 і 3 - задачі FAST.

3.6.2 Обробка подій

Механізм обробки подій дозволяє подіям, згенерованим різними джерелами, бути врахованими й обробленими настільки швидко, наскільки це можливо (наприклад, події від входів модуля 1, перевищення граничних значень на розрахункових модулях, і т.д).

Керуючі події

Вони є зовнішніми подіями, що можуть бути викликані:

• входами від 0 до 3 модуля 1 по передньому або задньому фронту;

• лічильними каналами модуля 1 (якщо він зконфігурований як лічильник);

• лічильними каналами (каналом) лічильних модулів,

• прийомом телеграм у ПЛК TSX 37-21/22, за допомогою модуля TSX FPP 20.

Коли канал має декілька джерел подій, дані, що модифікуються системою, використовуються для ідентифікації джерела, яке викликало подію. Можливо зконфігурувати до 8 подій у ПЛК TSX 37-10 і до 16 подій у ПЛК TSX 37-21/22; асоціація між каналом і номером події проводиться в екрані конфігурації каналів.

Поява такої події викликає переривання програми і призводить до виконання дій, що пов'язані з каналом I/O або прийомом телеграми (TSX 37-21/22), яка викликала подію.

Всі входи, пов'язані з каналом, який викликав подію, читаються автоматично*.

Читаються усі входи, оголошені користувачем у задачі EVTi.

Час обробки повинен бути мінімальним. Обновляються усі виходи, оголошені користувачем у задачі EVTi. Виходи, пов'язані з каналом, який викликав подію, повинні також бути оголошені, для наступного поновлення.

* Читання телеграм даних виконується функцією RCV_TLG.

Примітки:

I/O, оголошений у задачі EVTi, також зміниться в задачі MAST (у кожному періоді або циклі), що може викликати нерівномірності в хронології входів.

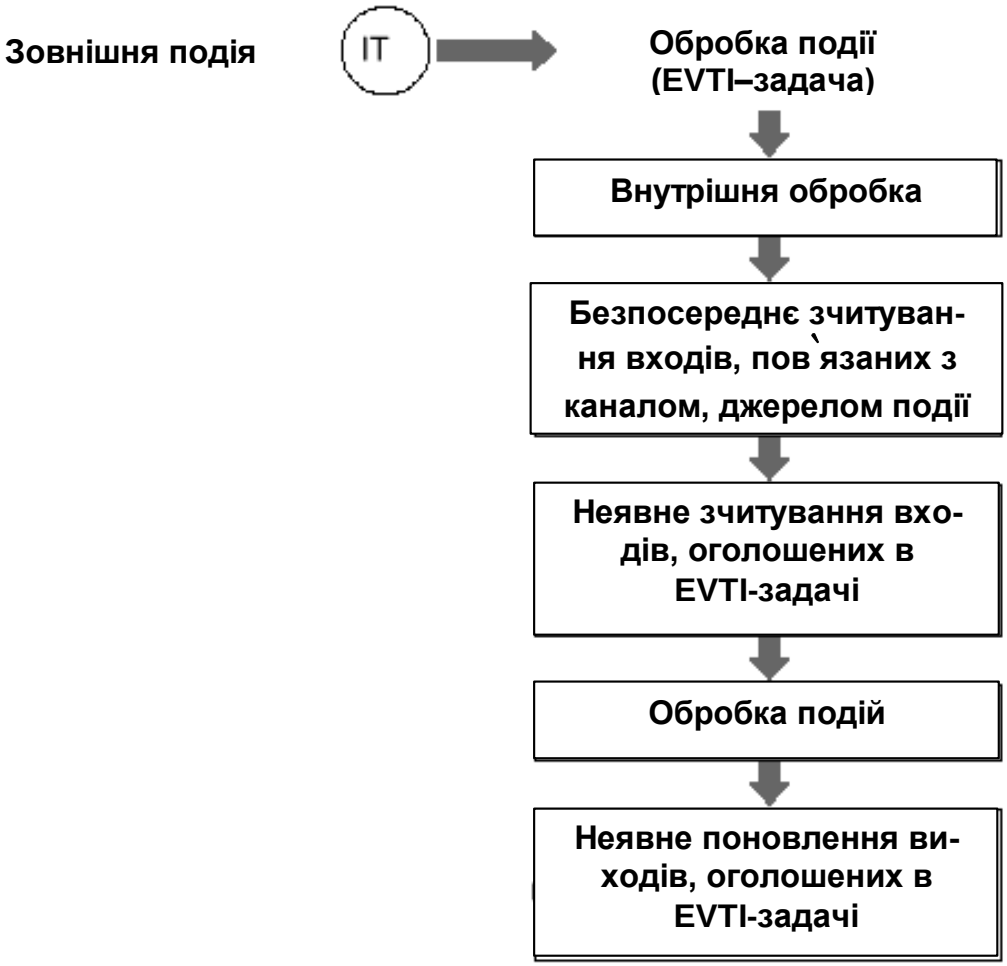


Рисунок 3.13 - Алгоритм обробки подій

Обробка подій може бути глобально дозволеною або забороненою прикладною програмою за допомогою системного біта %S38. Якщо одна або більша кількість подій відбувається, поки вони заборонені, обробка проводитись не буде.

Дві команди на мові PL7, доступні прикладній програмі, дозволяють глобальне маскування і розмаскування обробки подій. Якщо одна або більша кількість подій відбуваються, поки вони масковані, вони запам'ятовуються системою, і відповідна обробка буде виконуватися тільки після того, як вони будуть размасковані; порядок появи зберігається.

8 подій, можливих у ПЛК TSX 37-10 мають однаковий рівень пріоритету; з цієї причини обробка однієї події не може бути перервана іншою.

У ПЛК TSX 37-21/22 є 2 рівня пріоритету, для керуючих подій: подія 0 (EVT0) має більш високий пріоритет, ніж інші події (EVT1 до EVT15).

Коли подія відбувається, вона запам'ятовується в стеку, якщо обробляється подія з рівним або більш високим рівнем пріоритету. Обробка, пов'язана з цією подією, буде виконуватися тільки після обробки поточної.

Якщо стек переповнився, події будуть загублені; зазначена помилка встановлює системний битий %S39 у 1.

Примітки:

1. Аналогові вхідні модулі не повинні використовуватися в обробці події.

2. Читання/запис I/O, пов'язаних із задачею EVTi і оголошених користувачем, виконуються через канал (для лічильних модулів) або групи каналів (для дискретних модулів і для модулів аналогових виходів).

3. Для обробки кожної події можливо оголосити обміни максимум для 2-х вхідних модулів (перед обробкою події) і 2-х модулів виходів (після обробки події і будь-якого числа каналів або груп каналів).

4. Коли подія відбувається з ПЛК у стані STOP, процесор читає входи, модифікує виходи і збільшує системне слово %SW48, у якому ведеться підрахунок числа подій.

5. Будь-який обмін із входом або виходом у викликаної подією задачі може викликати втрату інформації про детектування фронтів, щодо опрацювання, виконаної по цьому каналу (або групі каналів), у задачі, де вони були оголошені: MAST або FAST.

3.7 Структура пам'яті користувача

Пам'ять ПЛК TSX 37-10/21/22 складається з 2 різних зон:

• Внутрішня пам'ять RAM, що використовується прикладною програмою і має ємність:

- 14 K слів для ПЛК TSX 37-10
- 20 K слів для ПЛК TSX 37-21/22.

Крім того, для ПЛК TSX 37-21/22 пам'ять програми може бути розширена до 32 K слів або 64 K слів (RAM або FLASH EPROM) картою PCMCIA.

• Пам'ять FLASH EPROM із 16 K слів, що служить як резервна пам'ять для:

- прикладної програми (тільки 15 K слів може використовуватися для резервування програми),
- внутрішніх слів %MW максимум 1000 слів (зарезервоване місце в 1 K слова).

Пам'ять програми може бути розділена на 5 зон пам'яті, фізично розділених між RAM, PCMCIA (якщо ПЛК TSX 37-21/22 обладнаний розширеною пам'яттю) і блоками пам'яті FLASH EPROM:

• Дані програми - завжди у внутрішній RAM.

• Прикладна програма (коментар і виконуваний код задачі) - у внутрішній RAM або на PCMCIA карті.

• Константи, початкові значення і конфігурація - у внутрішній RAM або на PCMCIA карті.

• Резервні копії прикладної програми, константи і дані конфігурації (якщо розмір прикладної програми менший, ніж 2.7 K інструкцій - у внутрішній пам'яті FLASH EPROM.

• Резервні копії максимум 1000 внутрішніх слів %MW у внутрішній пам'яті FLASH EPROM.

З урахуванням цих різних зон, можливі 2 типи організації пам'яті програми в залежності від оснащення ПЛК PCMCIA картою: прикладна програма знаходиться у внутрішній RAM або на PCMCIA карті (рисунок 3.14).

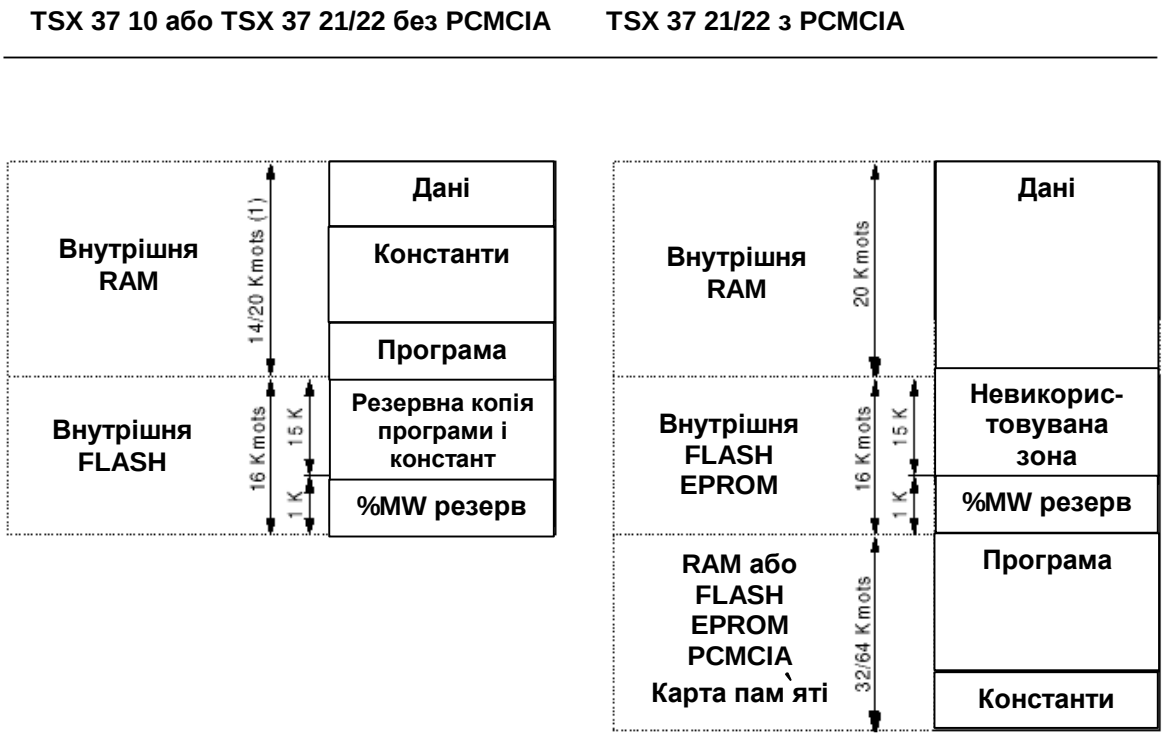


Рисунок 3.14 - Структура пам'яті

Дані: дані системи і програми.

Константи: слова-константи, початкові значення і конфігурація.

Програма: описувачі і здійснюваний код задачі.

Розміщення програми у внутрішній RAM.

Оскільки програма цілком завантажується в захищену внутрішню RAM* ПЛК (TSX 37-10 або TSX 37-21/22 без PCMCIA), її розмір повинен бути сумірний із розміром RAM:

• 14 K слів (TSX 37-10) розділені, наприклад, на 1 K слово даних програми і 13 K слів програми і констант.

• 20 K слів (TSX 37-21/22) розділені, наприклад, на 4 K слова даних програми і 16 K слів програми і констант.

Внутрішня пам'ять FLASH EPROM використовується для створення:

• резервних копій програми (код програми, константи, ініціалізовані значення, коментар, конфігурація),

• резервних значень внутрішніх слів %MW.

Резервна копія програми потребує, щоб ПЛК не мав PCMCIA карти і розмір програми був меншим або рівним 15 K слів. Програма, що займає 20 K слів у внутрішній RAM, не може мати резервні копії у внутрішній FLASH EPROM. Автоматичне переміщення програми з FLASH EPROM у RAM виконується, коли є втрата програми в RAM (через несправність або відсутність батареї). Ручне переміщення може також бути запитане через програмовний термінал.

У деяких випадках (помилка конфігурації, зміна програми, і т.д), може бути корисно очистити вміст RAM або FEPROM (FLASH EPROM) від програми. Для цього ПЛК повинен бути включений при утриманні кнопки DIAG мінімум на 10 секунд протягом самодіагностики.

Програма у PCMCIA карті (тільки TSX 37-21/22).

Коли пам'ять програми розширена PCMCIA картою, внутрішня пам'ять FLASH EPROM доступна тільки для резервної копії внутрішніх слів %MW.

Зробити резервну копію програми неможливо. У цьому випадку карта з пам'яттю містить повну програму (виконувана програма, константи, коментарі і конфігурація); внутрішня RAM зарезервована виняткова для даних (17.5 K слів). При розробці і налагодженні програми необхідно використовувати енергонезалежну RAM PCMCIA карти. Якщо програма експлуатаційна, вона може здійснюватися на тій же карті пам'яті або переміщена в FEPROM PCMCIA карти, що гарантує захист у випадку несправності батареї.

Примітка:

Коли програма сконфігурована для виконання у внутрішній пам'яті RAM ПЛК TSX 37-21/22 (карта з пам'яттю не була визначена в екрані конфігурації процесора), присутність цієї карти повинна спочатку бути оголошена в екрані конфігурації процесора перед пересилкою цієї програми в ПЛК, обладнаний PCMCIA картою пам'яті.

Захист програми.

При будь-якій структурі пам'яті ПЛК, розташована програма у внутрішній RAM або в PCMCIA карті може бути захищена, щоб заборонити доступ із PL7 (читання програми і налагодження) при

* Внутрішня RAM захищена необов'язковою батареєю на 3.6 V, що має ресурс 2 роки.

підключеному програмувальному терміналі. Щоб зняти захист такої програми, вона повинна бути переслана знову, без захисту, із термінала в ПЛК. Ця операція потребує вихідної програми, завантаженої в термінал.

Захищена програма у PCMCIA карті може бути виконана іншим ПЛК, але не може бути здубльована. PCMCIA карти мають защібку, що забороняє запис (тобто завантаження нової програми).

Перезавантаження зарезервованої програми.

На ПЛК TSX Micro можливо копіювати програму і константи із резервної карти пам'яті. Внутрішня пам'ять RAM може, таким чином, бути перезавантажена вмістом цієї резервної карти пам'яті.

Примітка: ця резервна функція недоступна, коли програма виконується в RAM або FLASH EPROM PCMCIA карти пам'яті.

Резервування програми, завантаженої у внутрішню RAM.

Ця операція дозволяє прикладну програму, що утримується у внутрішній RAM ПЛК, перемістити в PCMCIA карту резервної пам'яті.

3.8 Зберігання даних і прикладної програми

Щоб задовольнити вимогу зберігання даних у випадку відсутності або несправності батарейки процесора, ПЛК TSX 37 може використовуватися для копіювання 1000 внутрішніх слів %MW у внутрішню пам'ять FLASH EPROM. Це резервування внутрішніх слів %MW завжди буде пов'язано з резервною копією прикладної програми, коли вона знаходиться у внутрішній RAM і потребує деяких об'єктів мови програмування.

3.8.1 Використовувані об'єкти мови

Системне слово %SW96 управляє і/або проводить функції діагностики для зберігання/відновлення внутрішніх слів %MW і прикладної програми.

• **Біт 0**: Запит переміщення в резервну зону. Активний по передньому фронту. Він встановлюється в 0 системою, коли передній фронт виявлений.

• **Біт 1**: У стані 1 цей біт вказує, що функція зберігання завершилася. Він встановлюється в 0 системою, коли передній фронт виявлений.

• **Біт 2**: Звіт про зберігання 0 – зберігання без помилки 1 – помилка протягом зберігання.

• **Біти [3..5]**: Зарезервовані.

• **Біт 6**: Перевірка правильності резервної копії прикладної програми (функція та ж, що в системного біта %S96). 0 – резервна копія прикладної програми некоректна, 1 – резервна копія прикладної програми коректна.

• **Біт 7**: Перевірка правильності резервних значень %MW (функція та ж, що в системного біта %S97). 0 – %MW резервні значення некоректні, 1 – %MW резервні значення коректні.

• **Біти [8..15]**: Цей байт значущий тільки тоді, коли біт звіту (біт 2) - в 1 (помилка протягом зберігання).

1: числа %MW, що зберігаються більше зконфігурованого числа %MW.

2: числа %MW, що зберігаються більше 1000 або менше 0.

3: число %MW для відновлення більше від зконфігурованого числа %MW.

4: Розмір програми у внутрішній RAM більше 15 K слів (Нагадування: %MW завжди зберігаються, коли прикладна програма збережена у внутрішній FLASH EPROM).

5: Функції, заборонені в RUN.

6: Видалений резервний картридж із PLC.

7: Помилка запису в FLASH EPROM.

Системне слово %SW97 встановлює число %MW для зберігання.

Якщо значення - між 1 і 1000, прикладна програма, що утримується у внутрішній RAM і перші від 1 до 1000 %MW переміщуються у внутрішню пам'ять FLASH EPROM. Якщо значення рівне 0, тільки тоді прикладна програма, що утримується у внутрішній RAM переміщається у внутрішню пам'ять FLASH EPROM. Всі резервні значення %MW видаляться.

При “холодному” старті це слово ініціалізоване в 1, якщо внутрішня пам'ять FLASH EPROM не містить ніякого резерву %MW. Інакше воно ініціалізоване числом збережених слів.

Системний біт %S96 вказує результат перевірки правильності резервної копії прикладної програми 0 – резервна копія прикладної програми некоректна, 1 – резервна копія прикладної програми коректна

Цей біт може читатися в будь-який час (програмою в режимі настроювання) і особливо після “холодного” старту або “теплого” рестарту.

Системний біт %S97 вказує перевірку слушності резервних значень %MW: 0 – резервні значення %MW некоректні, 1 – резервні значення %MW коректні.

Цей біт може читатися в будь-який час (програмою в режимі настроювання) і особливо після “холодного” старту або “теплого” рестарту.

Вхід %I1.9: Цей дискретний вхід може бути зконфігурований як зовнішній вхід, щоб запросити переміщення RAM у внутрішню FLASH EPROM по передньому фронту.

3.8.2 Конфігурування методу зберігання

Користувач може виконувати зберігання, використовуючи два різних методи:

• використовуючи дискретний вхід %I1.9 (по передньому фронту). Функція цього входу конфігурується в екрані конфігурації процесора установкою прапорця “%I1.9 зберегти програму і перші %MWi”.

• з термінала настроювання установкою в “1” нульового біта системного слова %SW96.

Примітка:

Біт у позиції 0 системного слова %SW96 завжди активний. Вхід %I1.9 активний тільки тоді, коли він зконфігурований. Крім того, прапорець ”Скидання %MWi при холодному старті” в екрані конфігурації процесора не повинен бути відзначений.

3.8.3 Зберігання

Функція зберігання викликається по передньому фронту дискретного входу %I1.9 або коли біт 0 %SW96 зміниться з 0 у 1 у той час як ПЛК знаходиться в STOP. Число слів, що будуть переміщені, повинно бути попередньо встановлене в системному слові %SW97:

• **%SW97 = n** (n = від 1 до 1000): прикладна програма, утримувана в внутрішній RAM і перші n слів %MW переміщуються у внутрішню пам'ять FLASH EPROM.

• **%SW97 = 0**: тільки прикладна програма, утримувана у внутрішній RAM, переміщується у внутрішню пам'ять FLASH EPROM. Ця дія така ж, як і функція резервування програми в програмному забезпеченні PL7. У цьому випадку всі резервні значення %MW видалені.

Примітка: Коли вхідний %I1.9 зконфігурований як вхід зберігання, бажано зконфігурувати вхід %I1.8 як вхід RUN/STOP, щоб мати можливість зупинити ПЛК, без необхідності використання термінала.

Для операції зберігання є два можливих сценарія (рисунок 3.15).

Зберігання %MW видаляє будь-яку програму, уже записану в пам'яті FLASH EPROM. У результаті, якщо ця програма важлива користувачу, він повинен спочатку пересвідчитися, що в нього є резервна копія.

Наприкінці операції зберігання центральний дисплейний блок сигналізує OK або NOK відповідно до стана біта 2 %SW96 (звіт зберігання). Повідомлення OK або NOK обнуляється натисканням кнопки на центральному дисплеї або установкою ПЛК у RUN. Якщо живлення пропадає в процесі зберігання, то після його відновлення система виконує прозорий для користувача теплий перезапуск для закінчення операції.

Якщо холодний запуск виконується натисканням кнопки RESET або витягненням ручки на ПЛК TSX 37 2i, а операція зберігання не

закінчилася, це може призвести до втрати прикладної програми і даних, збережених у внутрішній пам'яті FLASH EPROM.

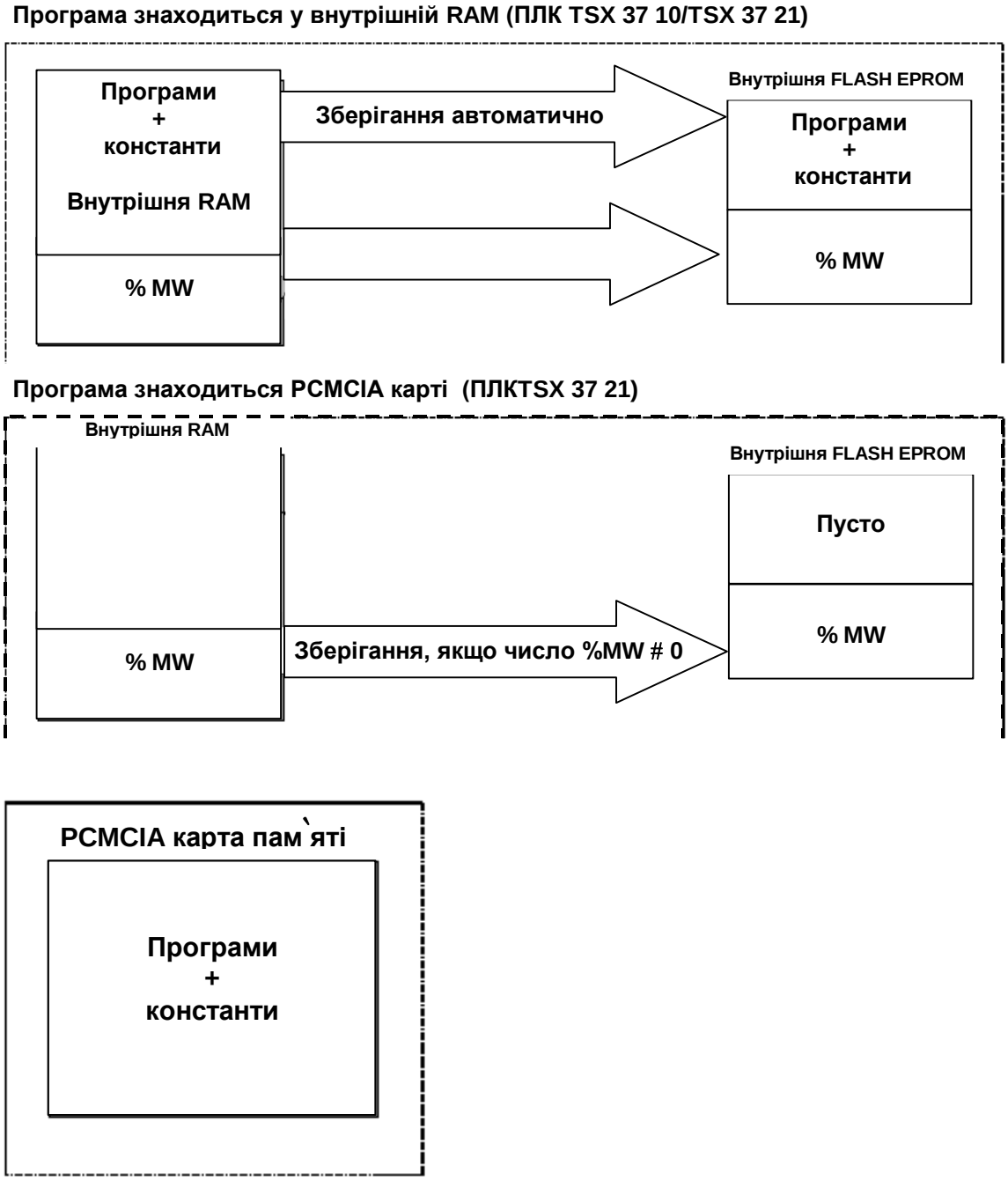


Рисунок 3.15 - Сценарії операції зберігання

3.9 Операції при збої або відновленні живлення

При збої живлення система запам'ятовує контекст програми і час збою, потім установлює всі виходи в 0.

Коли відновлюється живлення, збережений контекст порівнюється з поточною ситуацією; що визначає тип виконуваного старту:

• Якщо контекст програми змінився (втрата системного контексту або нова програма), ПЛК виконує холодний старт з ініціалізацією програми.

• Якщо програми ідентичні, ПЛК виконує теплий перезапуск без ініціалізації даних.

Якщо тривалість збою менше 10 мс для аналових сигналів або 1 мс для дискретних сигналів, це не помітно програмі, що виконується як звичайно.

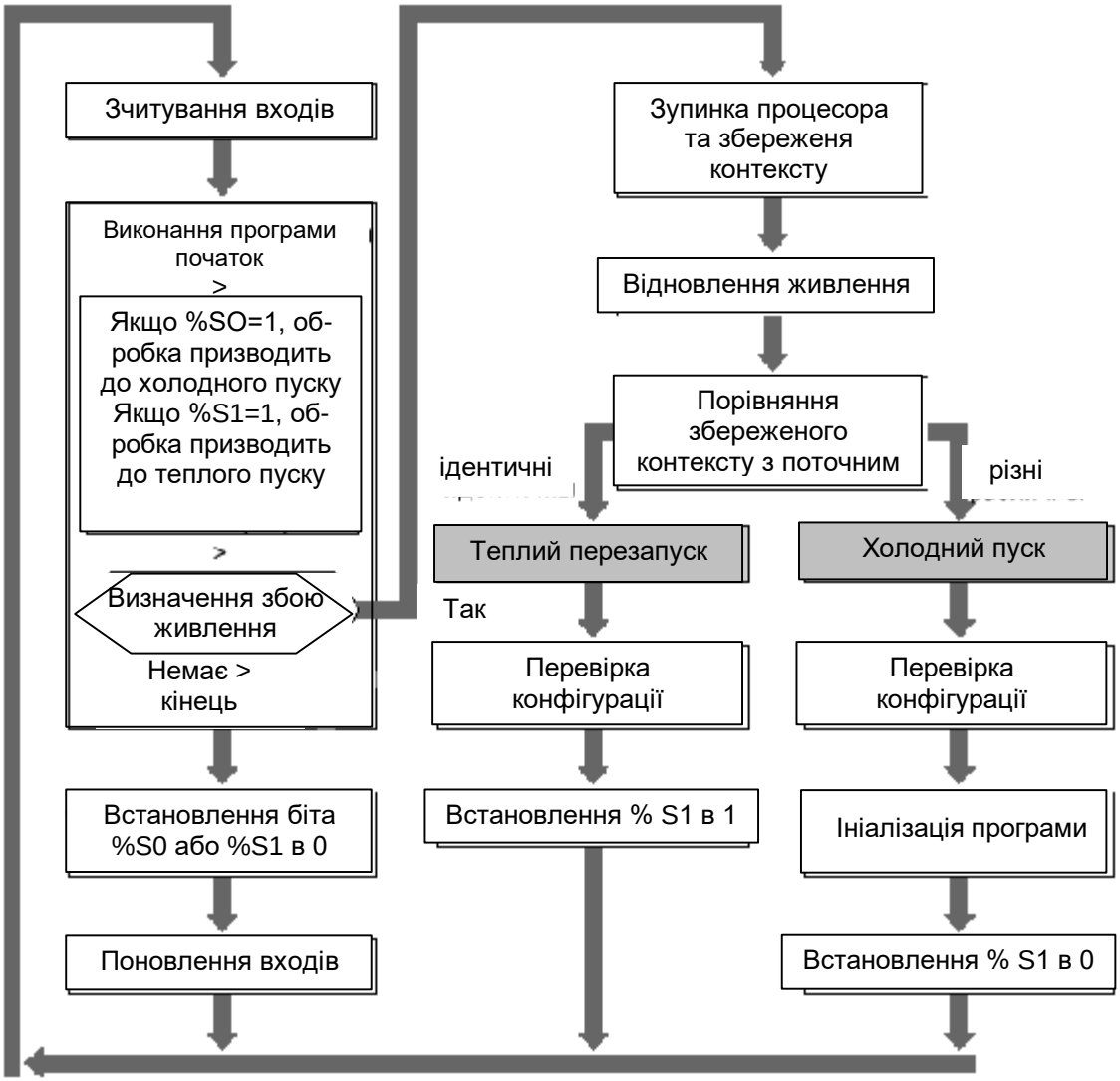


Рисунок 3.16 - Алгоритм роботи ПЛК при збої та відновленні живлення

3.10 Стандарт МЕК 1131.3

Починаючи зі створення перших програмованих контролерів, виробники протягом уже більше 25 років неперервно вдосконалюють апаратно-програмні характеристики своїх виробів. Багато хто з них створили унікальні комплекси програмного забезпечення, оригінальні мови програмування і могутню апаратну базу. Після першої ейфорії

користувачі, і насамперед великі промислові підприємства, почали усвідомлювати необхідність уніфікації засобів автоматизації. Застосування різних контролерів спричинило створення розвинутих служб з технічного обслуговування й експлуатації, що, безумовно, коштувало недешево для будь-якого підприємства. Більш того, підприємства були вже не в змозі керувати цим процесом (обслуговуванням систем АСУ) і все частіше стали звертатися до професійних організацій по впровадженню й обслуговуванню такої техніки. Уже наприкінці 70-их років необхідність стандартизації застосовуваних програмно-апаратних засобів стала життєвою вимогою і набула дуже критичний характер. У 1979 р. Міжнародна Електротехнічна Комісія (МЕК) засновує робочий комітет А65 зі створення стандарту для програмованих логічних контролерів. У розробці стандарту приймали також участь найбільші виробники програмованих контролерів Siemens, Alien Bradley, Modicon, General Electric, Schneider Electric та ін. На початку 90-их років з'явилися перші офіційні редакції цього стандарту:

МЕК 1131-1 - Загальні положення (1992).

МЕК 1131-2 - Специфікації й іспити устаткування (1992).

МЕК 1131-3 - Мови програмування (1993).

МЕК 1131-4 - Рекомендації користувачам.

МЕК 1131-5 - Специфікація сервісних служб повідомлень.

Вже в 1994 році користувачі побачили реалізації цього стандарту в контролерах (контролер класу нано - TSX Nano від Schneider Automation).

Отже, що собою представляє цей стандарт, що викликає усе більше питань у наших користувачів у частині мов програмування? Частина 3 "Мови програмовних логічних контролерів" визначає структуру й організацію прикладної програми в контролері, 5 мов програмування, типи змінних, адресацій і т.п. Для 5 мов програмування – **Ladder Diagram** (Діаграма сходової логіки), **Instruction List** (Список інструкцій), **Sequential Function Chart** (Граф послідовного керування), **Function Block Diagram** (Діаграма функціональних блоків) і **Structured Text** (Структурований текст), - визначені синтаксис, основні бібліотеки функціональних блоків, а також основні вимоги до редакторів цих мов. Розглянемо коротко основні можливості мов стандарту на прикладі програмного забезпечення від Schneider Automation (SA). Для контролерів серії Modicon TSX SA пропонує 2 пакети програмного забезпечення відповідного МЕК 1131.3 - **Concept** (з підтримкою всієї попередньої серії Modicon 984) і **PL7 Junior** (з підтримкою серії TSX 7).

PL7 Junior призначений для програмування контролерів Modicon TSX Micro і Modicon TSX Premium. У майбутньому передбачається злиття цього продукту з Concept і, таким чином, стане можливим програмування всієї гами Modicon TSX з єдиним програмним інструментарієм. У першій версії в цьому продукті реалізовані 4 мови за МЕК 1131.3. Це - LD, IL, ST і SFC. Ladder Diagram (LD) у графічному вигляді подає алгоритм вирішення

задач релейної логіки. Це одна з перших мов програмування контролерів, інтуїтивно зрозуміла будь-якому електротехніку і розрахована на вирішення задач послідовного керування. Проте, велика бібліотека функціональних блоків (обумовлених стандартом і додаткових) дозволяє вирішувати й інші задачі У цю бібліотеку входять лічильники, таймери, регістри, одновібратори, ПІД-регулятори, обчислювальні блоки, блоки комунікацій і операторського інтерфейсу, т.д. У вітчизняній техніці аналогом LD є мова релейно-контактних схем (РКС).

Мова Instruction List (IL) за функціональними можливостями є адекватною LD. Зовні схожа на асемблер, ця мова є більш звичною для професійних програмістів, вона дає також більше зручностей (у порівнянні з LD) при програмуванні складних обчислень.

Structured Text (ST) є Паскале-подібною мовою, що дозволяє реалізовувати складні алгоритми з програмуванням умов переходу, різних циклів, спеціальних математичних функцій і т.п.

І нарешті, Sequential Function Chart (SFC), давно відома багатьом користувачам за назвою Grafcet, представляє собою графічну мову для програмування роботи машин і установок, що мають, в основному, послідовний характер функціонування. Мова дуже зручна при налагодженні і діагностиці машин.

При орієнтації на стандарт МЕК 1131.3 усе-таки необхідна обережність, тому що цей стандарт не визначає деякі особливості, що можуть бути дуже важливими. Так, стандарт не визначає сервісні функції програмного забезпечення, такі як розвиненість і зручність налагоджувача, сервісні функції редакторів (копіювання, переноси, пошук і т.п.), наявність on-line help, рівень інтерфейса користувача, деякі важливі характеристики SFC, експорт/імпорт програм, конвертованість мов одна в одну та інше.

У цьому значенні PL7 Junior є досить сучасним і могутнім продуктом. Він пропонує сучасний інтерфейс користувача за стандартом Windows, завдяки чому користувачу вистачає лічених годин на початкове освоєння. Наявність багатовіконності дуже зручна при розробці і незамінна при налагодженні прикладної програми, а могутній on-line help дозволяє в лічені секунди видати довідку на будь-який запит користувача про функції системи. Могутній налагоджувач робить покрокове налагодження, користувач може вводити точки зупинки, форсувати біти і переглядати змінні в різних налагоджувальних таблицях як у символьному, так і в натуральному (адресному) вигляді. Разом із системою діагностики (користувач у цьому режимі може в графічному вигляді одержати не тільки вказівку на несправність, але і ймовірну причину збою) Налагоджувальник є дуже ефективним інструментарієм при введенні в експлуатацію і подальше обслуговування.

Переваги використання системи програмування, що відповідає стандарту МЕК 1131 очевидні. По-перше, користувачі одержують значну економію засобів при початковому навчанні інженерного й

обслуговуючого персоналу. Який би контролер не був обраний, користувач уже знає мови стандарту і не вимагає подальшого перенавчання. По-друге, завдяки визначеній стандартом, досить великій бібліотеці функціональних блоків скорочуються час і засоби при розробці прикладної програми. По-третє, спрощується також обслуговування АСУ ТП за рахунок знайомства зі стандартною структурою прикладної програми і принципами її роботи.

Список питань для самоконтролю

1. Наведіть основні принципи адресації каналів.
2. Як здійснюється адресація слів та бітів?
3. Що таке структуровані об’єкти? Назвіть їх види.
4. Що таке індексовані об’єкти? Наведіть приклади.
5. Основні принципи циклічного виконання програми.
6. Основні принципи періодичного виконання програми.
7. Яка структура програми ПЛК?
8. Як здійснюється багатозадачна обробка програми?
9. Наведіть основні характеристики керуючих задач.
10. Як здійснюється обробка подій в ПЛК?
11. Яка структура пам’яті в ПЛК?
12. Як здійснюється зберігання даних і прикладної програми?
13. Які операції здійснюються при збої та відновленні живлення?
14. Дайте загальну характеристику стандарту МЕК 1131.3.

4 МОВА ПРОГРАМУВАННЯ LADDER DIAGRAM

4.1 Основні поняття

Програми, написані мовою Ladder Diagram (LD), складаються із серії блоків, які виконуються послідовно ПЛК.

Блок складається з ряду графічних елементів, обмежених ліворуч і праворуч шинами живлення.

Вони представляють:

- I/O ПЛК (кнопки, сенсори, реле, індикаторні лампи і т.д).
- Функції Standard Control System (таймери, лічильники і т.д).
- Арифметичні, логічні оператори і оператори дій.
- Внутрішні змінні ПЛК.

Графічні елементи взаємозв’язані горизонтальними і вертикальними зв'язками (рис. 4.1).

Кожний блок складається з максимум 7 рядків і 11 стовпців, що розділені на 2 зони (рис. 4.2):

- Зона тестування (Test Zone), що містить умови, необхідні для виконання дій.
- Зона дії (Action Zone), що містить дії, які будуть виконані відповідно до результатів виконання зони тестування.

Кожен блок може бути ідентифікований міткою і озаглавлений коментарями.

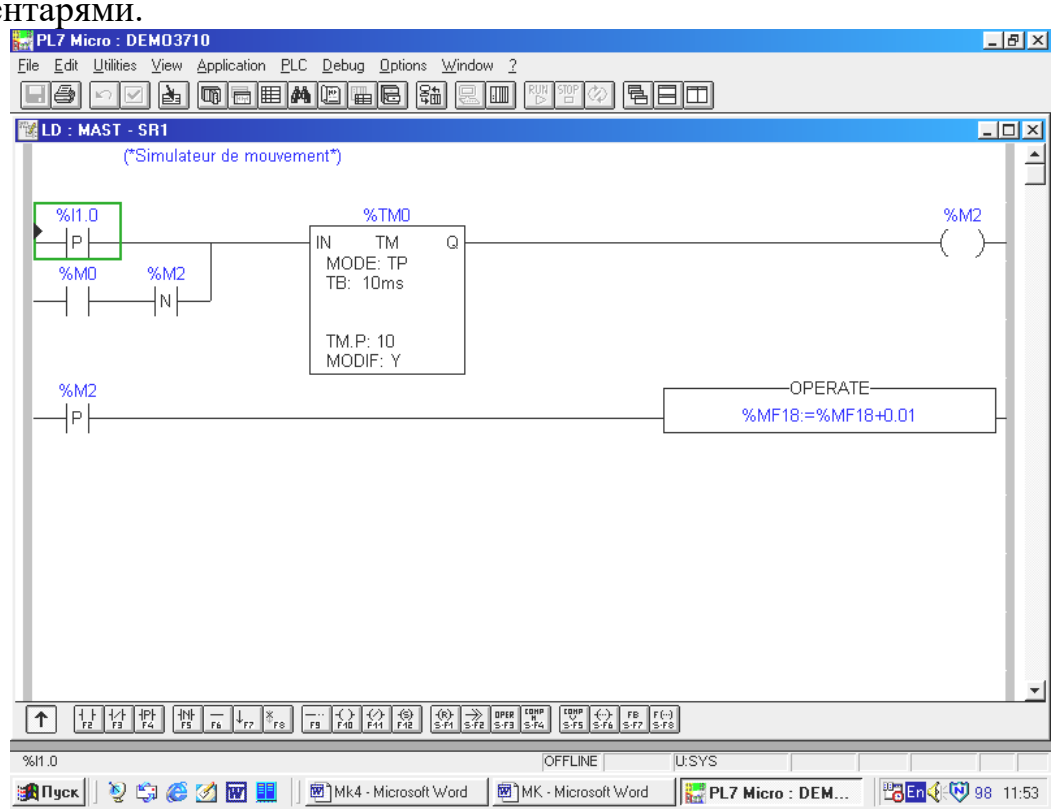


Рисунок 4.1 - Загальний вигляд програми на мові LD

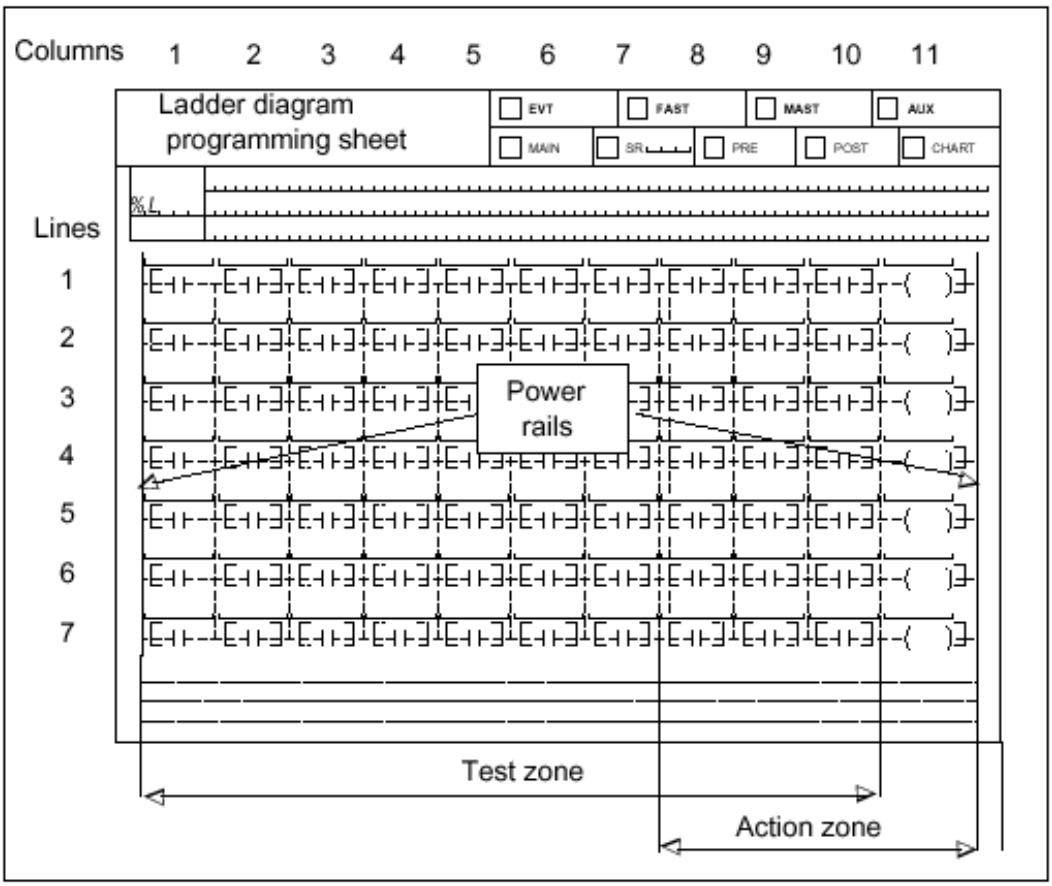
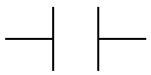
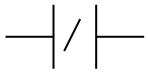
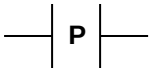
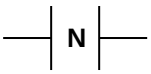


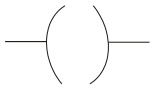
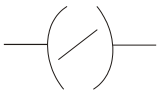
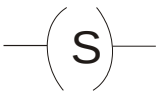
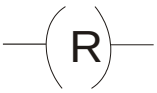
Рисунок 4.2 - Внутрішня структура блоку на мові LD

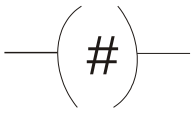
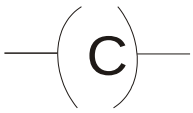
4.2 Графічні елементи

Основні елементи:

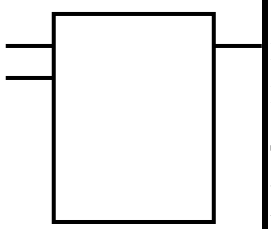
Основні елементи займають просту клітинку (висоту 1 рядка, ширину 1 стовпця).

Назва	Символ		Функція
Тестові елементи	Нормально відкритий контакт		Контакт закритий, коли біт контрольованого об'єкта = 1.
	Нормально закритий контакт		Контакт закритий, коли біт контрольованого об'єкта = 0.
	Контакти детектування фронту імпульсу		Передній фронт: контакт закритий, коли біт контрольованого об'єкта змінюється від 0 до 1.
			Задній фронт: контакт закритий, коли біт контрольованого об'єкта змінюється від 1 до 0.

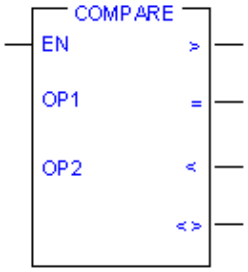
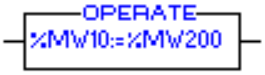
З'єднувальні елементи	Горизонтальні зв'язки	———	Горизонтальне з'єднання
	Вертикальні зв'язки	|	Вертикальне з'єднання
Елементи дій	Пряма котушка		Установка відповідного біта об'єкта у значення результату виконання тестової зони
	Інверсна котушка		Установка відповідного біта об'єкта у зворотнє значення результату виконання тестової зони
	Запірна котушка		Установка відповідного біта об'єкта в 1, коли результат виконання тестової зони = 1
	Відкривна котушка		Установка відповідного біта об'єкта в 0, коли результат виконання тестової зони = 1
	Умовний перехід до іншого блока	**-» %Li**	Дозволяє зв'язок з відповідним блоком. Перехід дійсний тільки в межах того ж самого об'єкта програмування (основна програма, підпрограма, і т.д). Якщо перехід активізується: • сканування поточного блока перерваний. • знаходиться відповідний помічений блок. • частина програми між блоком, де здійснюється перехід і позначеним блоком не виконується.

	Котушка умовного переходу		Використовується в мові Grafcet при програмуванні умов, пов'язаних з переходами на наступний крок
	Котушка виклику підпрограми		Дозволяє виклик підпрограми коли результат зони тестування – в 1. Якщо підпрограма викликана: • Перегляд поточного блока перервано. • Виконується підпрограма. • Перегляд поточного блока продовжується.
	Повернення з підпрограми	**<RETURN>**	Резервується для повертання з підпрограми, коли результат виконання тестової зони = 1
	Зупинка програми	**<HALT>**	Виконання зупинки програми, коли результат виконання тестової зони = 1

Функціональні блоки:

Назва	Символ		Функція
Тестові елементи	Блоки: 1. Таймер 2. Лічильник 3. Одновібратор 4. Регістр 5. DRUM - контролер		Кожний з стандартних функціональних блоків використовує входи/виходи, які дозволяють їм бути зв'язаними з іншими графічними елементами.

Блоки операцій:

Назва	Символ		Функція
Тестові елементи	Вертикальний блок порівняння		Дозволяє порівняння двох операндів. В залежності від результату, відповідний вихід змінюється на 1 Розмір: 2 стовпці/4 рядки
	Горизонтальний блок порівняння		Дозволяє порівняння двох операндів. Вихід змінюється на 1, коли результат порівняння позитивний. (Блок може містити до 4096 характеристик). Розмір: 2 стовпці/1 рядок.
Елементи дій	Блок дій		Виконує арифметичні, логічні операції і т.д, і використовує синтаксис мови Structured Text. (Блок може містити до 4096 характеристик). Розмір: 4 стовпці/1 рядок.

4.3 Структура блока

4.3.1 Мітки

Мітки використовують для ідентифікації блока в межах об'єкта програми (основна програма, підпрограма, і т.д), але вони не обов'язкові (рисунок 4.3).

Мітки мають синтаксис **%Li** (i = 0...999) і зафіксовані в лівій вершині перед шиною живлення.

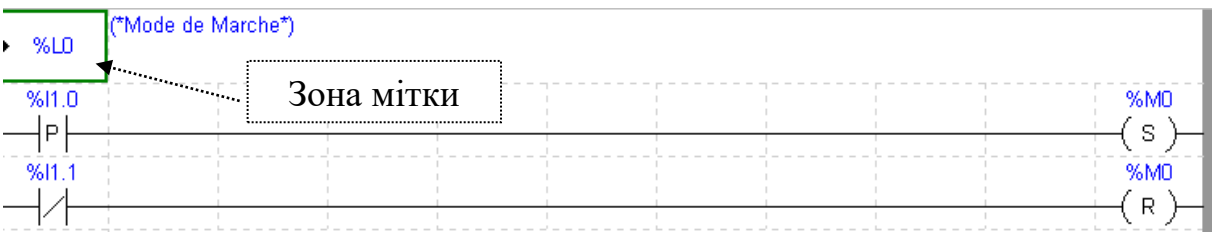


Рисунок 4.3 - Зона мітки

Кожна мітка може бути призначена тільки на один блок у межах того ж самого об'єкта програми.

Система переглядає блоки в порядку, в якому вони були введені, незалежно від порядку номерів мітки.

4.3.2 Коментарі

Коментарі інтегровані у блок і містять до 222 буквено-цифрових знаків, обмежених з обох кінців знаками (* і *).

Коментарі відображені в зарезервованій зоні у верхній частині блока (рисунок 4.4).

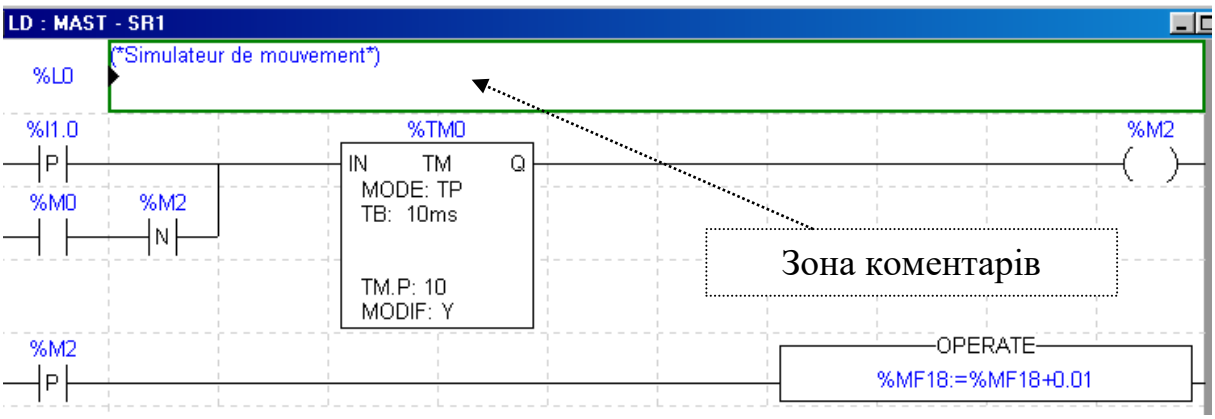


Рисунок 4.4 - Зона коментарів

Коментарі зберігаються в ПЛК, і користувач може завжди одержати до них доступ. Тому вони використовують пам'ять програми.

4.3.3 Блоки

Представлення блока подібно релейно-контактній схемі автоматики (рисунок 4.5).

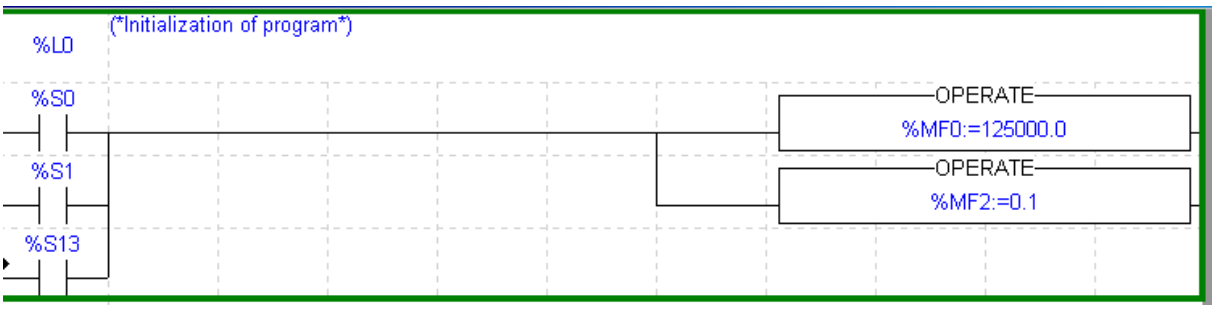


Рисунок 4.5 - Представлення блока в LD

Прості графічні елементи займають одиночну лінію і стовпець у межах блока.

Усі рядки контактів починаються на лівій шині живлення і повинні закінчитися на правій шині живлення.

Тести завжди зафіксовані на стовпцях від 1 до 10. Дії завжди зафіксовані на 11 стовпці.

Напрямок струму такий:

• для горизонтальних зв'язків - **зліва направо;**

• для вертикальних зв'язків - **в обох напрямках.**

Зона тестування містить елементи тестування:

- Прямі, інвертовані, детектування фронту імпульсу.
- Вертикальні та горизонтальні блоки порівняння.
- Функціональні блоки.

Зона дії містить:

- Елементи дій (прямі, інверсні, котушки запирання, відмикання, котушки переходу).
- Блоки дій.

4.4 Керування в програмі на мові LD

4.4.1 Прості кроки керування

Керування котушкою створює умови для зміни стану контакту (рисунок 4.6, а). Використовують до 10 контактів послідовно на одному рядку (рисунок 4.6, б). Максимум 7 контактів може тестуватися паралельно в одному стовпці і керувати 7-ма котушками, розміщеними паралельно (рисунок 4.6, в).

4.4.2 Кроки керування, що використовують кілька рядків контактів

Крок може бути розділений на кілька незалежних рядків контактів, які керуються незалежними котушками (рисунок 4.7).

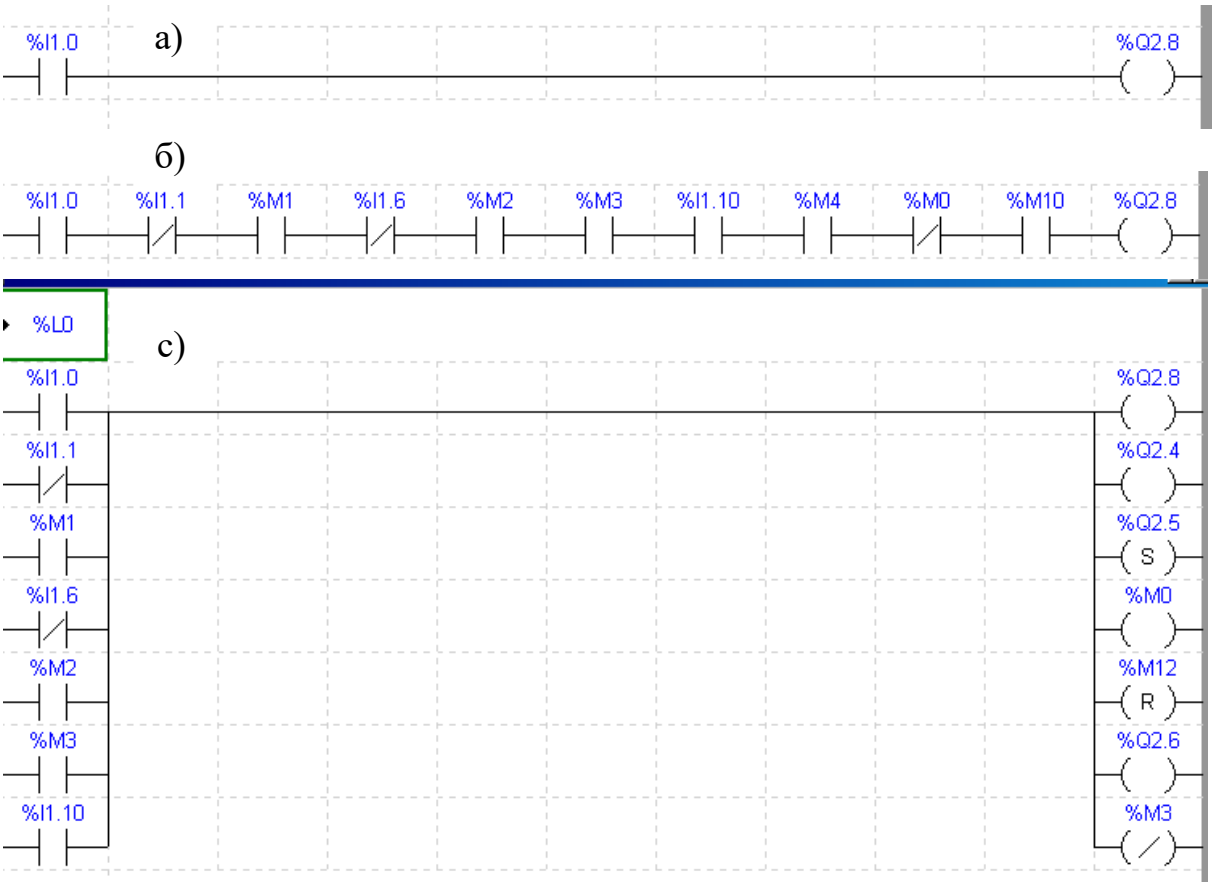


Рисунок 4.6 - Прості кроки керування

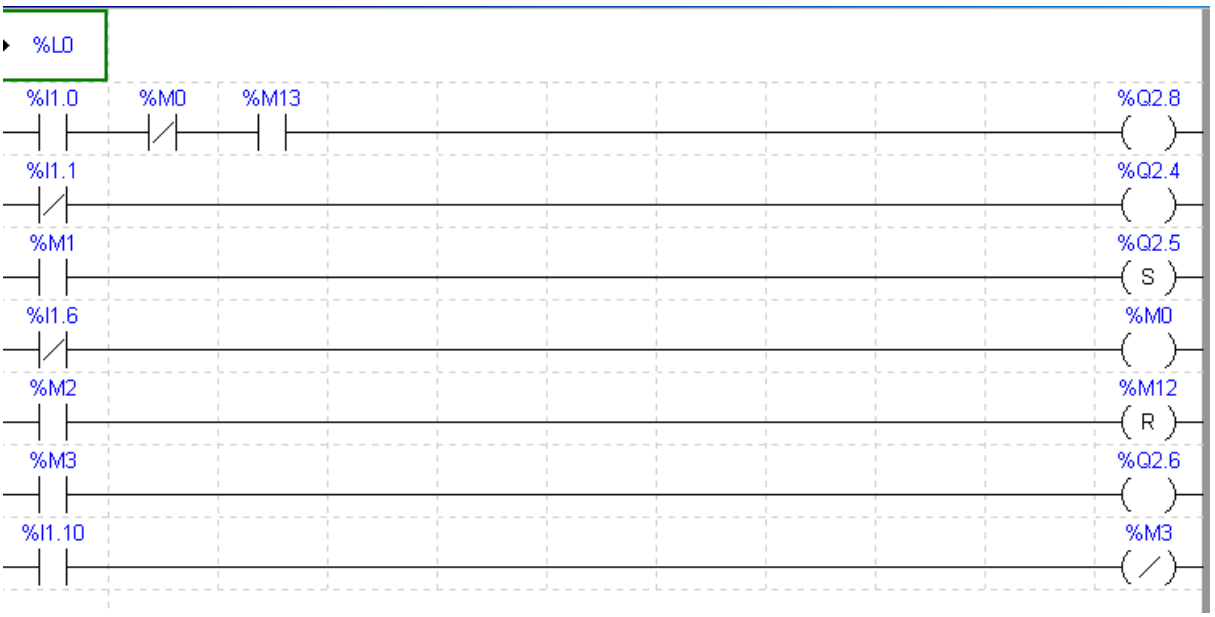


Рисунок 4.7 - Сім незалежних рядків контактів

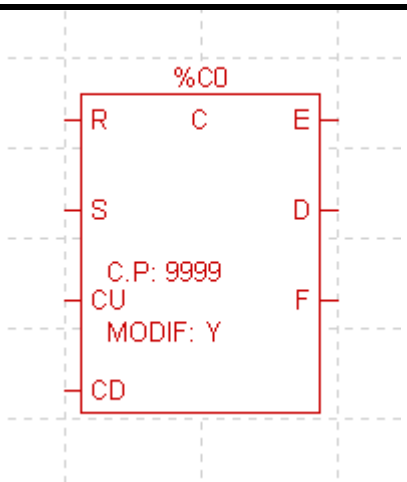
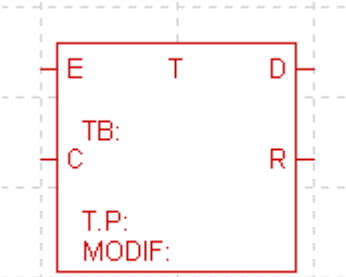
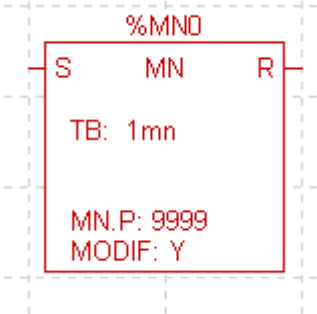
4.4.3 Кроки керування з функціональними блоками

До функціональних блоків належать (таблиця 4.1): реверсивні лічильники (Counter); таймери (Timer); таймери послідовності імпульсів

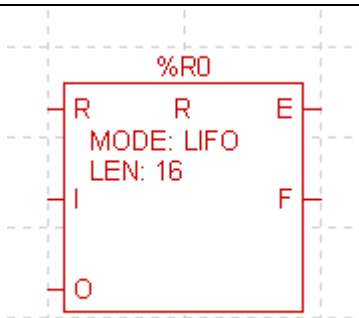
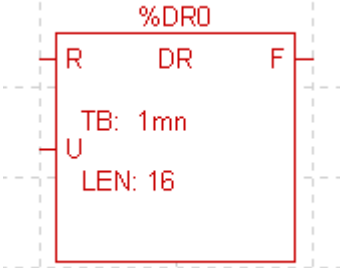
(Series Timer); одновібратори (Monostable); регістри (Register); DRUM-контролер (Drum)

Всі функціональні блоки розміщуються в зоні тестування.

Таблиця 4.1 - Функціональні блоки

Назва функціонального блоку	Позначення
Реверсивний лічильник (займає 4 рядки і 2 стовпці)	
Таймер (займає 3 рядки і 2 стовпці)	
Таймер послідовності імпульсів (займає 3 рядки і 2 стовпці)	
Одновібратор (займає 3 рядки і 2 стовпці)	

Таблиця 4.1 - Продовження

Регістр (займає 3 рядки і 2 стовпці)	
DRUM-контролер (займає 3 рядки і 2 стовпці)	

Фунціональні блоки можуть бути розташовані послідовно (рисунок 4.8).

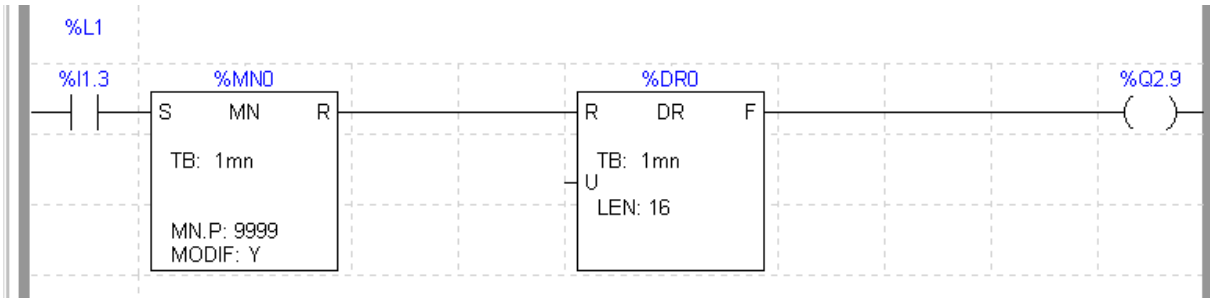


Рисунок 4.8 - Послідовне розміщення функціональних блоків

4.4.4 Кроки керування з блоками порівняння та блоками дії

Горизонтальний блок порівняння (рисунок 4.9, а) займає 1 рядок і 2 стовпця і порівнює 2 операнди OP1 і OP2. Вихід блока встановлюється в 1, коли умова порівняння задовольняється.

Вертикальний блок порівняння (рис. 4.9, б) займає 4 рядка і 2 стовпця і порівнює 2 операнди OP1 і OP2. Відповідний вихід блока встановлюється в 1, коли відповідна умова порівняння задовольняється.

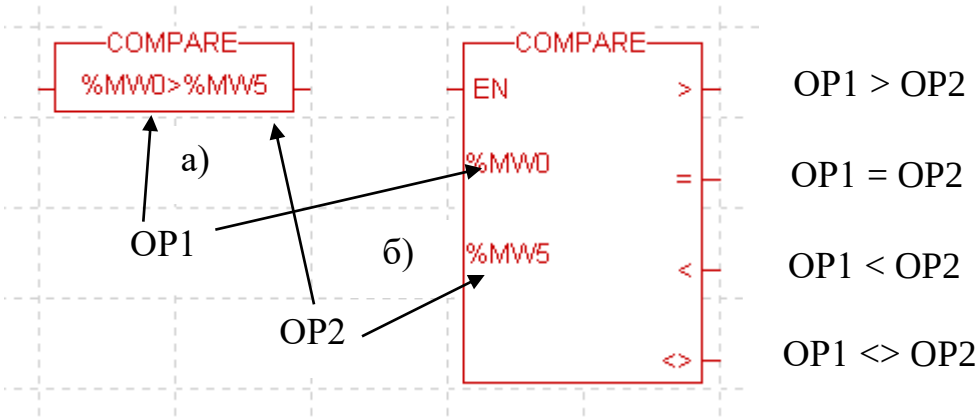


Рисунок 4.9 - Приклади блоків порівняння

В операційному блоці (рисунок 4.10), який займає 1 рядок і 4 стовпці, проводяться арифметичні, логічні та інші обчислення, виклик функцій.

Операційні блоки завжди зафіксовані в зоні дії. Вони написані мовою Structured Text і завжди прямо зв'язуються з правою шиною живлення.

Рисунок 4.10 - Операційний блок

Функціональні блоки і операційні блоки можуть бути змішані (рисунок 4.11).

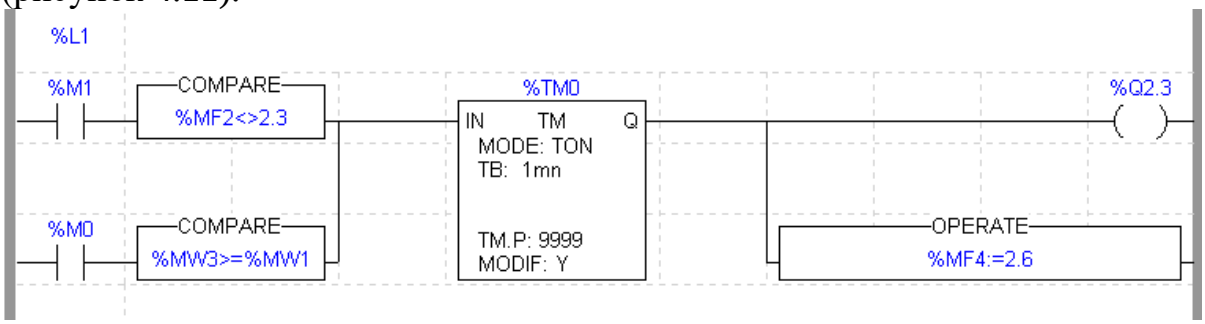


Рисунок 4.11 - Комбінація функціональних блоків і операторів дії

4.5 Правила для виконання кроків керування

Кроки керування виконуються крок за кроком зліва направо.

Приклад кроків керування (рисунок 4.12) містить графічні елементи, які мають горизонтальні і вертикальні зв'язки (крім шини живлення), але незалежні від інших графічних елементів (немає вертикальних зв'язків між кроками).

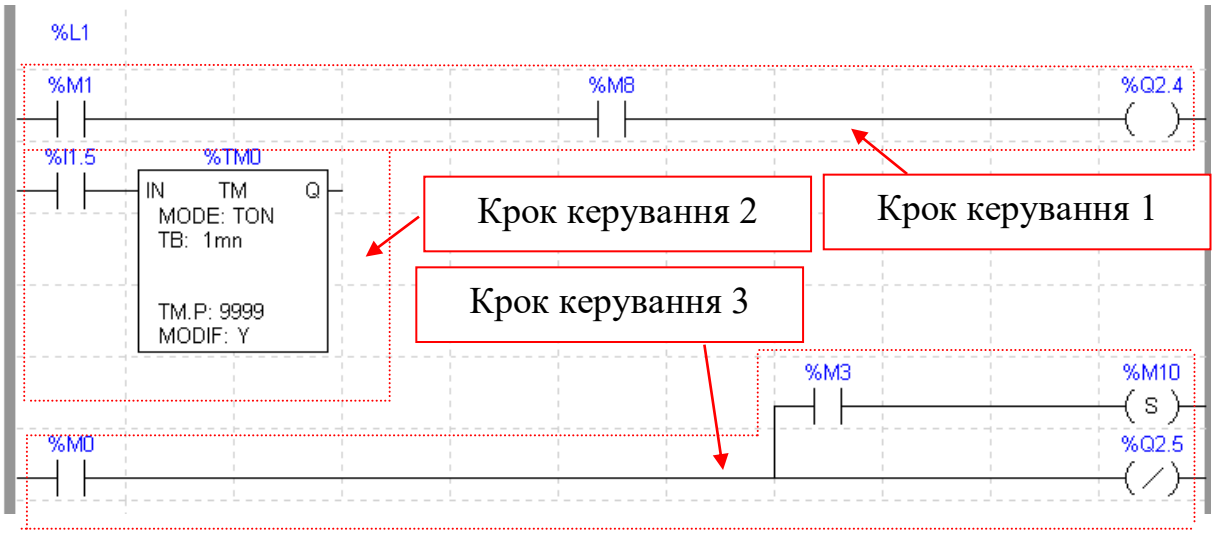


Рисунок 4.12 - Приклад кроків керування

Кроків керування 1 у верхньому лівому кутку - перший крок, який буде виконано.

Кроки виконуються таким чином: від верху до низу, рядок за рядком, і в кожнім рядку зліва направо.

У випадках, де є вертикальне з'єднання, кроки підпорядковані тій самій логіці.

Під час виконання система ПЛК:

- оцінює логічний стан кожного контакту, відповідно до поточного значення внутрішніх об'єктів і стани входів-виходів.
- виконує операції з обробки, пов'язані з функціями, функціональними блоками і підпрограмами.
- модифікує біти об'єктів, пов'язані зі котушками, виходи модулів I/O.
- йде до іншого позначеного кроку керування в тому ж самому програмному модулі (перестрибує до іншого кроку ->> **%Li**), повертається до викличного модуля **<RETURN>**, чи зупиняє програму **<HALT>**.

Дії керування на кроці (рисунок 4.13) виконуються в такому порядку:

- обчислення кроку до 1-ої вертикальної збіжності зв'язку, контакти A, B і C;
- обчислення 1-го підкроку, контакт D;
- обчислення кроку до 2-ї вертикальної збіжності зв'язку, контакт E;
- обчислення 2-го підкроку, контакти F і G;
- обчислення котушки H.

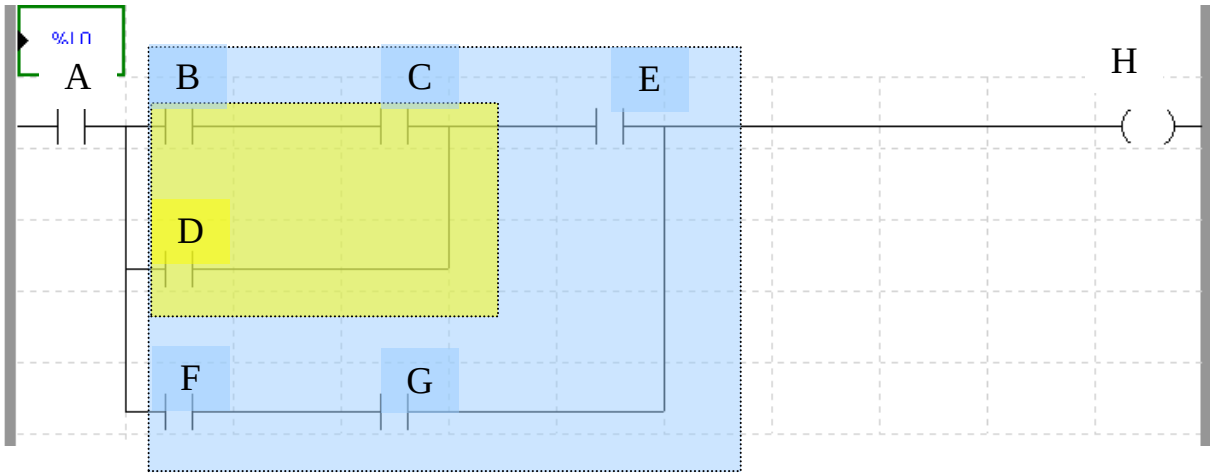


Рисунок 4.13 - Приклад обчислення кроків керування

Список питань для самоконтролю

1. Які основні поняття мови Ladder Diagram?
2. Назвіть тестові елементи Ladder Diagram.
3. Назвіть з’єднувальні елементи Ladder Diagram.
4. Назвіть елементи дій Ladder Diagram.
5. Які блоки операцій використовуються в Ladder Diagram?
6. Як використовуються мітки в Ladder Diagram?
7. Як використовуються коментарі в Ladder Diagram?
8. Як здійснюється керування в Ladder Diagram?
9. Які функціональні блоки використовуються в Ladder Diagram?
10. Як працюють блоки порівняння?
11. Які правила для виконання кроків керування в Ladder Diagram?

5 МОВА ПРОГРАМУВАННЯ INSTRUCTION LIST

5.1 Основні поняття

Програма, написана мовою Instruction List (IL) (рисунок 5.1), складається із серії команд, які виконуються послідовно ПЛК.

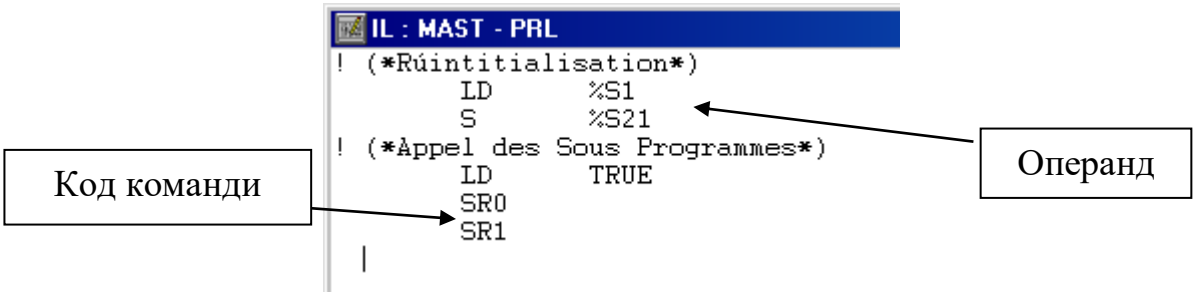


Рисунок 5.1 - Приклад програми на мові IL

Приклад команди: LD %I1.0.

Кожна команда складається з коду команди і операнда.
Ці команди діють на:

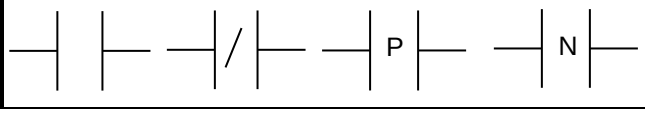
- I/O ПЛК (кнопки, детектори, реле, сигнальні вогні і т.д);
- Функції Standard Control System (таймери, лічильники і т.д);
- Арифметичні і логічні операції, операції переносу;
- Внутрішні змінні ПЛК.

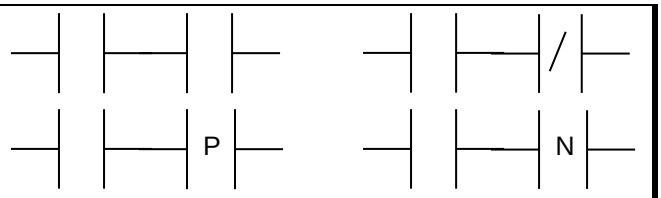
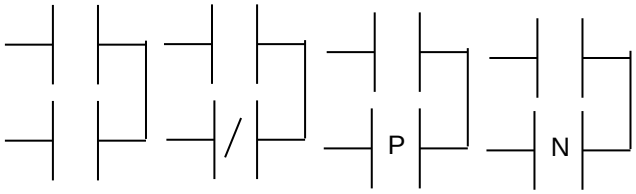
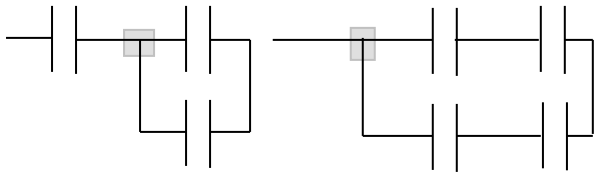
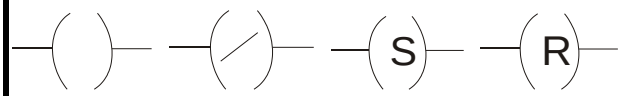
Є два типи команд:

- Команди тестування, які містять умови, необхідні для виконання дій: LD, AND, OR і т.д.;
- Команди дій, які перевіряють результат тестового циклу: ST, STN, R, і т.д.

5.2 Команди

Основні команди:

Назва	Команди	Еквівалентні функції
Тестові команди	LD, LDN, LDR, LDF	

Тестові команди	AND, ANDN, ANDR, ANDF	
	OR, ORN, ORR, ORF	
	AND(, OR((8 рівнів круглих дужок)	
	XOR, XORN, XORR, XORF	Ексклюзивне OR
	MPS MRD MPP	
	N	Заперечення
Команди дій	ST, STN, S, R	
	JMP, JMPC, JMPCN	Використовується для переходу (безумовного, умовного на булевий результат 1 чи умовного на булевий результат 0) до позначеної команди, або входу чи виходу
	SRn, RET, RETC, RETCN	Використовується для переходу на початок підпрограми. Повернення з підпрограми (безумовне, умовне з булевим результатом 1 чи умовне з булевим результатом 0).

	END, ENDC, ENDCN, HALT	Кінець програми (безумовний, умовний з булевим результатом 1 чи умовний з булевим результатом 0). Виконання програми зупиняється.

Команди функціональних блоків:

Назва	Команди	Функції
Тестові елементи	Блоки: Таймер Лічильник Одновібратор Регістр DRUM-контролер	Існують команди керування кожним із блоків. Форма використання - пряме приєднання I/O функціональних блоків.

Числові команди:

Назва	Команди	Функції
Тестові елементи	LD[.....] AND[.....] OR[.....] Приклад: LD[%MW10<1000]	Використовуються для порівняння двох операндів. Вихід змінюється на 1, коли результат = 1. Результат дорівнює 1, коли %MW10<1000.
Елементи дій	[.....] Приклад: [%MW10:=%MW0+10]	Виконує арифметичні, логічні операції і т.д. Використовується синтаксис мови Structured Text Результат операції %MW0 + 10 поміщений у внутрішню %MW10.

5.3 Структура програми

5.3.1 Основні правила

Подібно мові Ladder Diagram, команди організовані в послідовність команд (еквівалентно до кроку), і називаються епізодом. Кожна послідовність складається з однієї чи більшої кількості тестових команд. Результат виконання цих команд використовується до однієї чи більшої кількості команд дій.

Команда займає однин рядок.

Кожен епізод починається точкою виклику (згенерованою автоматично). Вона може включати коментарі і бути ідентифікованою міткою. Приклад:

```
!       (*Waiting for drying*)
        %L2:
        LD      %I0.1
        AND     %M10
        ST      %Q2.5
```

5.3.2 Коментарі

Коментарі можуть займати до 3 рядків (тобто 222 буквено-цифрових знаки), обмежених з обох кінців знаками (* і *). Вони полегшують тлумачення частини програми, до якої вони належать, але є не обов'язкові.

Коментарі відображаються тільки з першого рядка програми.

Якщо частина програми вилучається, зв'язані з нею коментарі, також вилучаються.

Коментарі зберігаються в ПЛК, і користувач може звертатися до них завжди. Тому вони використовують пам'ять програми.

5.3.3 Мітки

Мітки використовуються для ідентифікації епізоду в об'єкті програми (основна програма, підпрограма, і т.д) але є не обов'язкові.

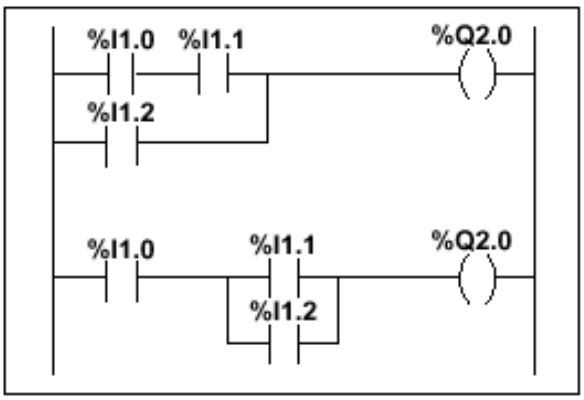
Мітки мають синтаксис **%Li** (i=0...999) і зафіксовані в початковій стадії епізоду.

Мітка може тільки бути призначена на одиночний епізод у межах того ж самого об'єкта програми.

Операційна система ПЛК переглядає епізоди в порядку, у якому вони були введені, незалежно від порядку номерів мітки.

5.3.4 Використання дужок

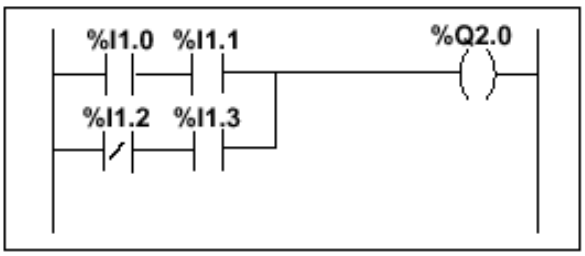
Можна використовувати круглі дужки з командами AND чи OR. Ліва кругла дужка пов'язана з командою AND чи OR. Права кругла дужка представляє команду і обов'язкова для кожної лівої дужки відкриття.




```
LD      %I1.0
AND     %I1.1
OR      %I1.2
ST      %Q2.0

LD      %I1.0
AND(    %I1.1
OR      %I1.2
)
ST      %Q2.0
```

Рисунок 5.2 - Приклад команди AND(

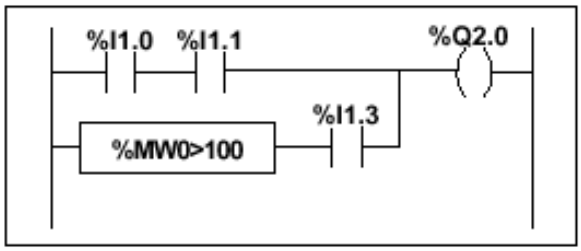



```
LD      %I1.0
AND     %I1.1
OR(N    %I1.2
AND     %I1.3
)
ST      %Q2.0
```

Рисунок 5.3 - Приклад команди OR(

Наступні оператори можуть бути пов'язані з круглими дужками:

- N (заперечення), приклад: AND(N чи OR(N;
- F (задній фронт), приклад: ANB(F чи OR(F;
- R (передній фронт), приклад: AND(R чи OR(R;
- [(порівняння).




```
LD      %I1.0
AND     %I1.1
OR(     [%MW0>100]
AND     %I1.3
)
ST      %Q2.0
```

Рисунок 5.4 - Приклад команди OR([…]

Програма може мати до восьми рівнів вкладених круглих дужок.

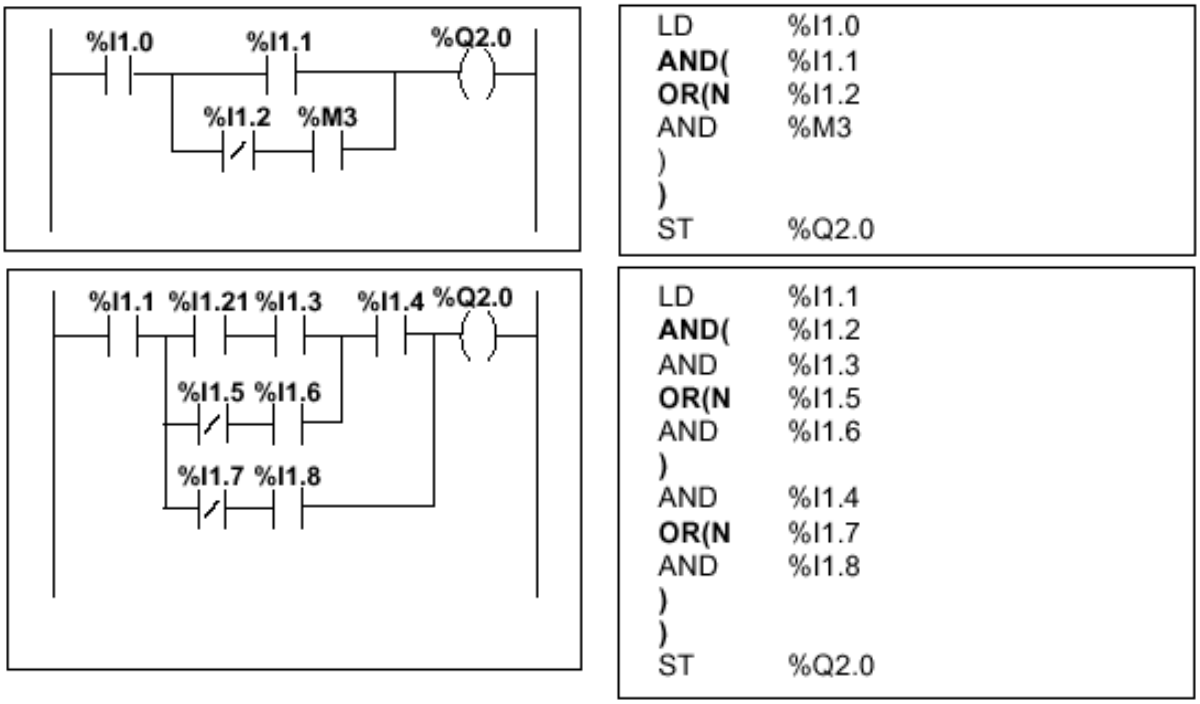


Рисунок 5.5 - Приклади команд із вкладеними круглими дужками

Примітки:

• Кожна кругла дужка відкриття (тобто ліва) **повинна** послідовно закриватися круглою дужкою (правою).

• Мітка **%Li** не повинна бути розміщена у виразах між круглими дужками. Це також застосовується до команд переходу JMP і команд виклику підпрограм SRi.

• Команди ST, STN, S і R не повинні бути запрограмовані між двома круглими дужками.

5.3.5 Команди MPS, MRD і MPP

Ці три типи команд використовуються, для обробки розподілу до котушок.

Вони використовують буфер, відомий як стек, який здатний зберігати до 3 логічних бітів даних.

Команда **MPS** *(Memory PuSH)* зберігає результат останньої тестової команди наверху стека, і зрушує інші значення в напрямку до дна стека.

Команда **MRD** *(Memory Read)* читає вершину стека.

Команда **MPP** *(Memory Pop)* читає вершину стека, і зрушує інші значення в напрямку до вершини стека.

Ці команди не можуть використовуватися у виразах між круглими дужками.

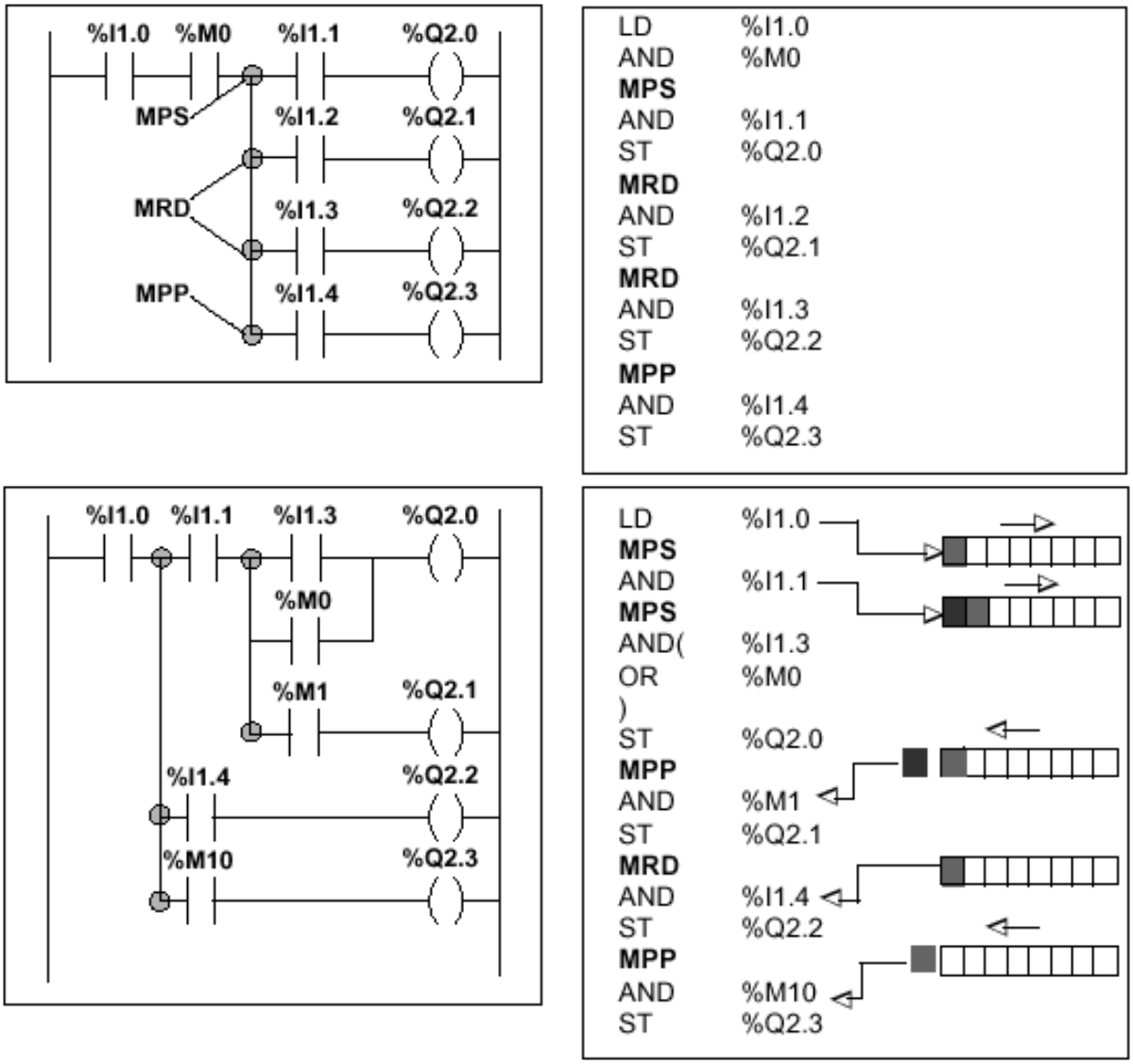


Рисунок 5.6 - Приклади команд MPS, MRD і MPP

5.3.6 Принцип програмування функціональних блоків

Функціональні блоки системи керування можуть бути запрограмовані двома різними шляхами:

• з командами, визначеними до кожного функціонального блока (приклад: CU %Ci). Це найпростіший і найбільш прямий шлях;

• з командами BLK блокової структури OUT_BLK і END_BLK.

Принцип прямого програмування

Команди керують введеннями блоків (приклад: CU).

До виходів можна звертатися у формі бітів (приклад: % C8.D).

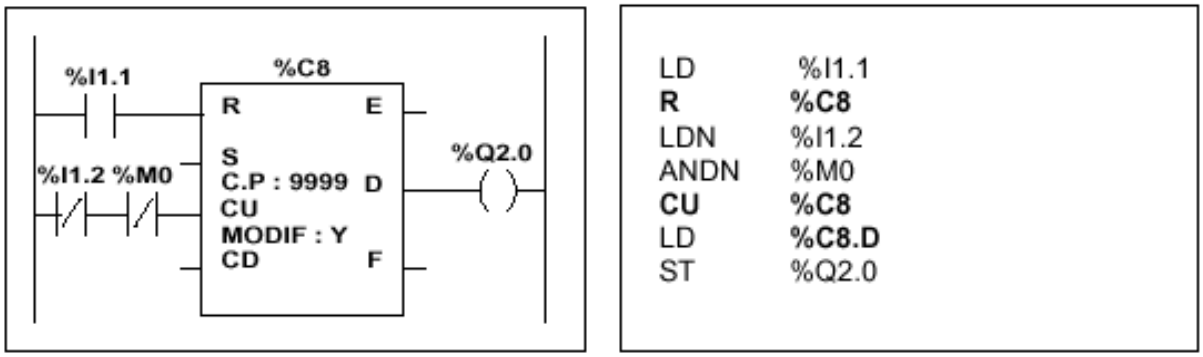


Рисунок 5.7 - Приклад до принципу прямого програмування

Принцип структурного програмування

Цей тип програмування використовує послідовність команд, побудованих наступними командами:

- **BLK** вказує початок блока.
- **OUT_BLK** використовується для прямо з'єднаних виходів блока.
- **END_BLK** вказує кінець блока.

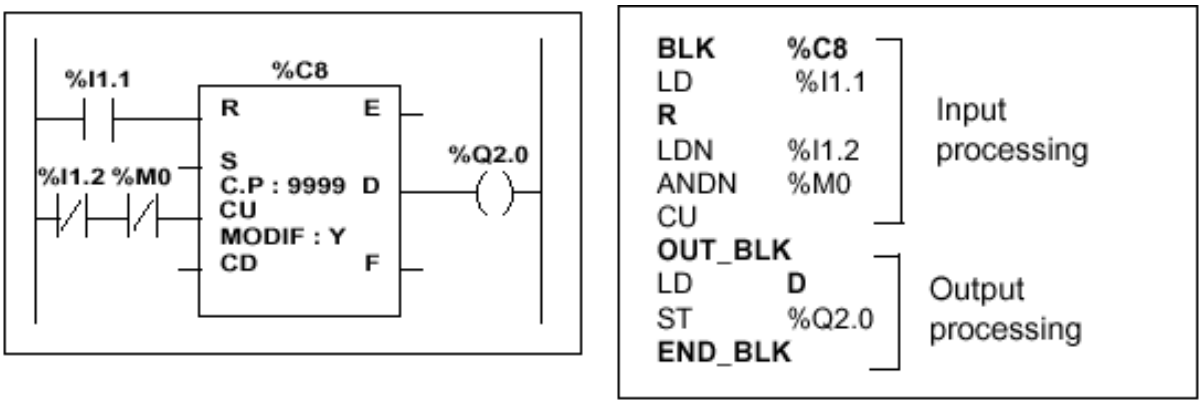


Рисунок 5.8 - Приклад до принципу структурного програмування

Структурне програмування вимагає додаткових команд BLK, OUT_BLK і END_BLK, і тому вимагає більшої кількості пам'яті у порівнянні з безпосереднім програмуванням.

5.4 Правила для програм, що виконуються на Instruction List

Програма IL виконується послідовно команда за командою. Перша команда в серії команд повинна завжди бути або команда LD, або безумовна команда (наприклад: JMP).

Усі команди (крім LD і безумовних команд) використовують логічний результат виконання попередньої команди.

Круглі дужки можуть використовуватися, для зміни порядку, у якому приймаються логічні результати.

Приклади:

AND %I1.2 Логічний результат – AND попереднього логічного результату і стан біта %I1.2.

OR(%M10 Логічний результат – стан біта %M10.
) Логічний результат – OR попередній логічний результат і логічний результат команди, зафіксованої перед командою з круглими дужками.

ST %Q2.0 %Q2.0 – стан попереднього логічного результату.

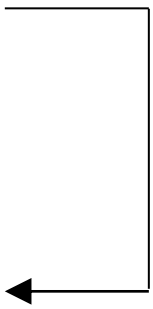
AND %M0 Логічний результат – AND попереднього логічного результату і стан біта %M0.

LD %I1.1 Логічний результат – стан біта %I1.1

Послідовне виконання команд може змінюватися переходом (JMP) і командами виклику підпрограми.

Приклад:

```
!       LD      %M0
        JMPC    %L10
!       LD      %I1.1
        AND     %M10
        ST      %Q2.0
!       %L10:
        LD      %I1.3
        AND     %M20
        …….
```



Перехід до мітки %L10 якщо %M0=1

Список питань для самоконтролю

1. Які основні поняття мови Instruction List?
2. Назвіть тестові команди Instruction List.
3. Назвіть команди дій Instruction List.
4. Назвіть команди функціональних блоків Instruction List.
5. Назвіть числові команди Instruction List.
6. Які основні правила програмування Instruction List.
7. Як використовуються коментарі в Instruction List?
8. Як використовуються мітки в Instruction List?
9. Як використовуються дужки в Instruction List?
10. Як працюють команди MPS, MRD, MPP?
11. Охарактеризуйте принципи програмування функціональних блоків в Instruction List.

6 МОВА ПРОГРАМУВАННЯ STRUCTURED TEXT

6.1 Основні поняття

Мова Structured Text (ST) використовується для створення програм, написаних із програмними рядками, створених з буквено-цифрових знаків.

Твердження ST складає основну частину мови ST, і серія тверджень використовується для визначення програми.

Головні команди в мові Structured Text такі:

• розрядні команди,

• арифметичні і логічні команди на слова і подвійні слова,

• арифметичні команди на числа з плаваючою точкою,

• чисельне порівняння на слова, подвійні слова і числа з плаваючою точкою,

• чисельні конверсії,

• команди на біт, слово, подвійне слово і таблиці чисел з плаваючою точкою,

• команди салінгових символів,

• алфавітно-цифровий компаратор,

• команди адміністрації часу,

• програмні команди,

• команди керування,

• команди стандартного функціонального блока,

• точні команди обміну,

• специфічні для програми команди (зв'язок, ПІД-керування, і т.д).

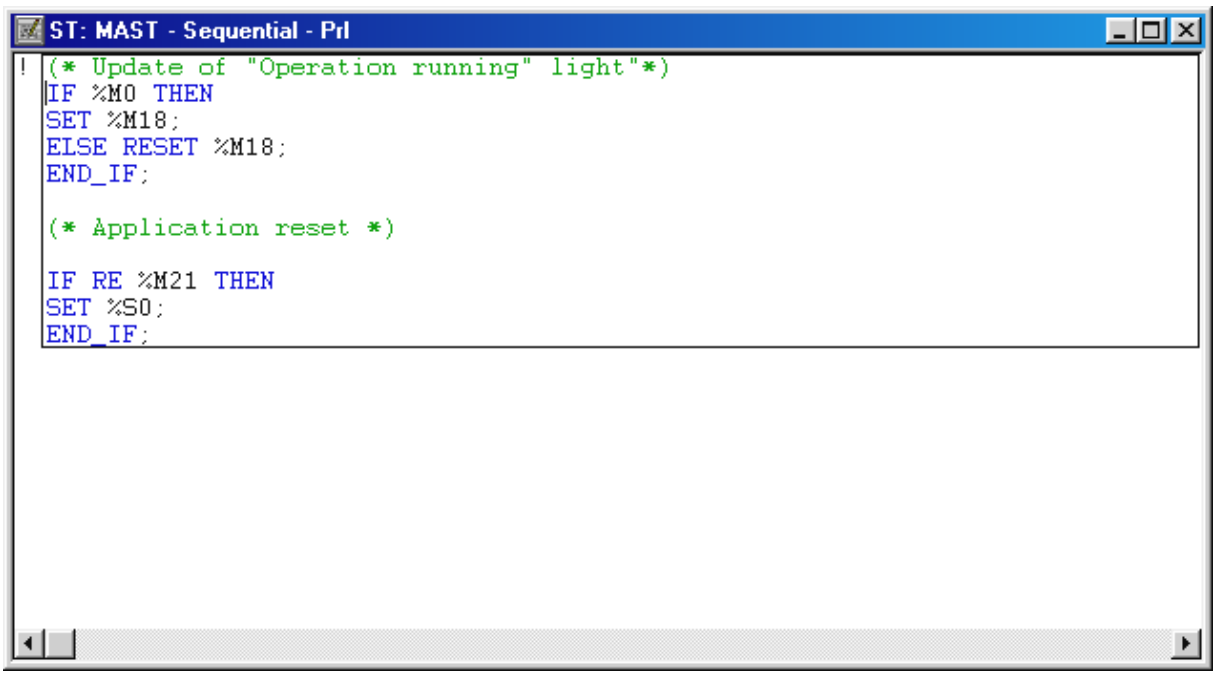


Рисунок 6.1 - Приклад програми на мові ST

6.2 Команди

Розрядні команди:

Назва	Функція
:=	Розрядне присвоєння
OR	Логічний OR
AND	Логічний AND
XOR	Ексклюзивне логічне OR
NOT	Інверсія
RE	Передній фронт імпульсу
FE	Задній фронт імпульсу
SET	Установка в 1
RESET	Установка в 0

Чисельне порівняння для слів, подвійних слів і чисел з плаваючою точкою:

Назва	Функція
<	Менше ніж
>	Більше ніж
<=	Менше ніж чи рівний чомусь
>=	Більше ніж чи рівний чомусь
=	Рівний чомусь
<>	Відмінний від

Разрядові таблиці:

Назва	Функція
Table := Table	Присвоєння між двома таблицями
Table := Word	Присвоєння слова таблиці
Word := Table	Присвоєння таблиці слову
Table := Double word	Присвоєння подвійного слова таблиці
Double word := Table	Присвоєння таблиці до подвійному слову
COPY_BIT	Копіювання розрядної таблиці в розрядну таблицю
AND_ARX	AND між двома таблицями
OR_ARX	OR між двома таблицями
XOR_ARX	Ексклюзивне OR між двома таблицями
NOT_ARX	Заперечення таблиці
BIT_W	Копіювання розрядної таблиці в таблицю слова
BIT_D	Копіювання розрядної таблиці в таблицю подвійного слова
W_BIT	Копіювання таблиці слова в розрядну таблицю
D_BIT	Копіювання таблиці подвійного слова в розрядну таблицю

Цілочисельна арифметика для слів і подвійних слів:

Назва	Функція
+	Додавання
-	Віднімання
*	Множення
/	Цілочисельне ділення
REM	Залишкове значення від цілочисельного ділення
SQRT	Цілочисельний квадратний корінь
ABS	Абсолютне значення
INC	Збільшення
DEC	Зменшення

Арифметика для чисел з плаваючою точкою:

Назва	Функція
+	Додавання
-	Віднімання
*	Множення
/	Ділення
SQRT	Квадратний корінь
ABS	Абсолютне значення

Логічні команди для слів і подвійних слів:

Назва	Функція
AND	Логічний AND
OR	Логічне OR
XOR	Ексклюзивне логічне OR
NOT	Логічне доповнення
SHL	Логічний зсув ліворуч
SHR	Логічний зсув праворуч
ROL	Логічний поворот ліворуч
ROR	Логічний поворот праворуч

Команди перетворення чисел:

Назва	Функція
BCD_TO_INT	Код BCD → Ціле число
INT_TO_BCD	Ціле число → Код BCD
GRAY_TO_INT	Код GRAY → Ціле число
INT_TO_REAL	Ціле число → Число з плаваючою точкою
DINT_TO_REAL	32-розрядне ціле число → Число з плаваючою точкою
REAL_TO_INT	Число з плаваючою точкою → Ціле число
REAL_TO_DINT	Число з плаваючою точкою → 32-розрядне ціле число
DBCD_TO_DINT	32-розрядний код BCD → 32-розрядне ціле число
DINT_TO_DBCD	32-розрядне ціле число → 32-розрядний код BCD
DBCD_TO_INT	32-розрядний код BCD → 16-розрядне ціле число
INT_TO_DBCD	16-розрядне ціле число → 32-розрядний код BCD

Команди для таблиць слів і подвійних слів:

Назва	Функція
Table := Table	Присвоєння між двома таблицями
Table := Word	Ініціалізація таблиці
+, -, *, /, REM	Арифметичні операції між таблицями
+, -, *, /, REM	Арифметичні операції між виразами і таблицями
EQUAL	Порівняння двох таблиць
SUM	Логічне доповнення таблиці
NOT	Інверсія таблиці
AND, OR, XOR	Логічні операції між двома таблицями
AND, OR, XOR	Логічні операції між виразами і таблицями
FIND_EQW, FIND_EQD	Знаходження першого елемента, рівного значенню
FIND_GTW, FIND_GTD	Знаходження першого елемента, більшого ніж значення
FIND_LTW, FIND_LTD	Знаходження першого елемента, меншого ніж значення
MAX_ARW, MAX_ARD	Знаходження максимального значення в таблиці
MIN_ARW, MIN_ARD	Знаходження мінімального значення в таблиці
OCCUR_ARW, OCCUR_ARD	Порядок місцезнаходження значення в таблиці
SORT_ARW, SORT_ARD	Сортування таблиці в порядку зростання чи спадання
ROL_ARW, ROL_ARD	Поворот таблиці наліво
ROR_ARW, ROR_ARD	Поворот таблиці направо

Команди для таблиці чисел з плаваючою точкою:

Назва	Функція
Table := Table Table := Floating point	Присвоєння між двома таблицями Ініціалізація таблиці

Команди для символьних рядків:

Назва	Функція
STRING_TO_INT	ASCII → Ціле число
STRING_TO_DINT	ASCII → Подвійне ціле число
INT_TO_STRING	Ціле число → ASCII
DINT_TO_STRING	Подвійне ціле число → ASCII
STRING_TO_REAL	ASCII → Число з плаваючою точкою
REAL_TO_STRING	Число з плаваючою точкою → ASCII
<, >, <=, >=, =, <>	Алфавітно-цифрове порівняння
FIND	Позиція підрядка у рядку символів
EQUAL_STR	Позиція першого різного символу
LEN	Довжина рядка символів
MID	Витяг підрядка із рядка символів
INSERT	Вставка підрядка в рядок символів
DELETE	Видалення підрядка з рядка символів
CONCAT	Зв'язати два рядки
REPLACE	Замінити рядок
LEFT	Початок рядка
RIGHT	Закінчення рядка

Програмні команди:

Назва	Функція
HALT	Виконання зупинки програми
JUMP	Перехід до мітки
Sri	Запит підпрограми
RETURN	Повернення з підпрограми
MASKEVT	Події маскування в PLC
UNMASKEVT	Відкриття маски події в PLC

Команди адміністрації часу:

Назва	Функція
RRTC	Читає дату системи
WRTC	Змінює дату системи
PTC	Читає дату і зупиняє код
ADD_TOD	Додає інтервал часу до часу дня
ADD_DT	Додає інтервал часу до дати і часу
DELTA_TOD	Різниця між вказаними годинами
DELTA_D	Різниця між вказаними датами (без врахування годин)
DELTA_DT	Різниця між вказаними датами (із врахуванням годин)
SUB_TOD	Обчислення інтервалу часу від задоної години
SUB_DT	Обчислення інтервалу часу від заданих дати і години
DAY_OF_WEEK	Читає поточний день тижня
TRANS_TIME	Перетворює тривалість часу до дати
DATE_TO_STRING	Перетворює дату до рядка символів
TOD_TO_STRING	Перетворює час до рядка символів
DT_TO_STRING	Перетворює завершену дату до рядка символів
TIME_TO_STRING	Перетворює тривалість до рядка символів

6.3 Структура програми

6.3.1 Основні принципи

Програма Structured Text організована у твердженнях. Кожні твердження ST складаються з таких елементів:

• мітка,
• коментарі,
• команди.

Кожний з цих елементів необов'язковий, тобто можливе існування порожнього твердження, твердження, що складається тільки з коментарів чи складається тільки з мітки. Кожне твердження починається з мітки виклику, яка генерується автоматично.

Приклад:

```
!   %L2:   (* Твердження з міткою, коментарі *)
             SET %M0; %MW4 := %MW2 + %MW9;
               (* і кілька команд *)
              %MF12 := SQRT (%MF14);
```

6.3.2 Коментарі

Коментар обмежений з обох кінців символами (* і *), він може бути поміщений в будь-якому місці у твердженні. Немає ніякого обмеження на число коментарів у твердженні. Він полегшує тлумачення твердження, для якого призначений, але є необов'язковим.

• Будь-які символи можна використовувати в коментарі.
• Число символів обмежене 256 на коментар.
• Вкладені коментарі не дозволяються.
• Коментар може бути кілька рядків у довжину.

Коментарі зберігаються в ПЛК і користувач може звертатися до них у будь-який момент. Через це вони споживають пам'ять програми.

6.3.3 Мітки

Мітка використовується для виклику твердження в об'єкті програми (основна програма, підпрограма, і т.д) але є необов'язковою.

Мітка має такий синтаксис: **%Li**, де i=0...999 і зафіксована на початку твердження. Виклик мітки може бути здійснений тільки на одиночне твердження в межах того ж самого об'єкта програми (SR, головна програма, програмний модуль).

Мітки можуть бути розташовані в будь-якому порядку, залежно від того, як твердження введені і прийняті системою протягом сканування.

6.3.4 Структури керування в програмі

Програма складається з команд. Твердження ST може містити кілька команд. Кожна команда повинна закінчуватися знаком “;”.

Існує чотири структури керування:

• умовна дія IF,

• умовні повторювані дії WHILE і REPEAT,

• повторювана дія FOR.

Кожна структура керування включена між ключовими словами, і вона починається і закінчується в тому самому твердженні. Можна вкласти структури керування одна в одну, незалежно від їхнього типу.

Умовна дія IF... END_IF;

Проста форма - команда виконує дію, якщо умова правдива (рисунок 6.2).

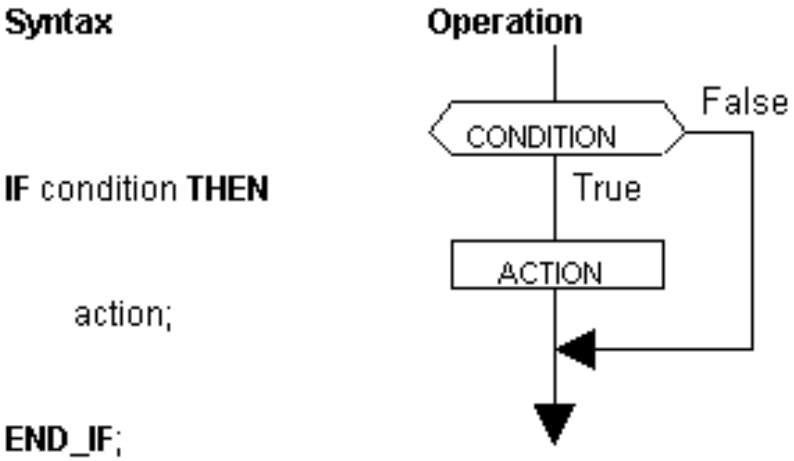


Рисунок 6.2 - Проста форма умовної дії IF... END_IF;

Приклад простої форми умовної дії IF... END_IF; :

```
IF %M0 AND %M12 THEN
    RESET %M0;
    INC %MW4;
    %MW10:=%MW8+%MW9;
END_IF;
```

Основна форма:

- Умови можуть бути множинними.
- Кожна дія представлена як список команд.
- Кілька "IF" структур керування можуть бути вкладеними.
- Немає ніякого обмеження на число команд ELSIF.
- Є максимум одна частина ELSE.

```
IF condition 1 THEN

      action1;
      ELSIF condition 2 THEN
            action2;
      ELSE action3;
END_IF;
```

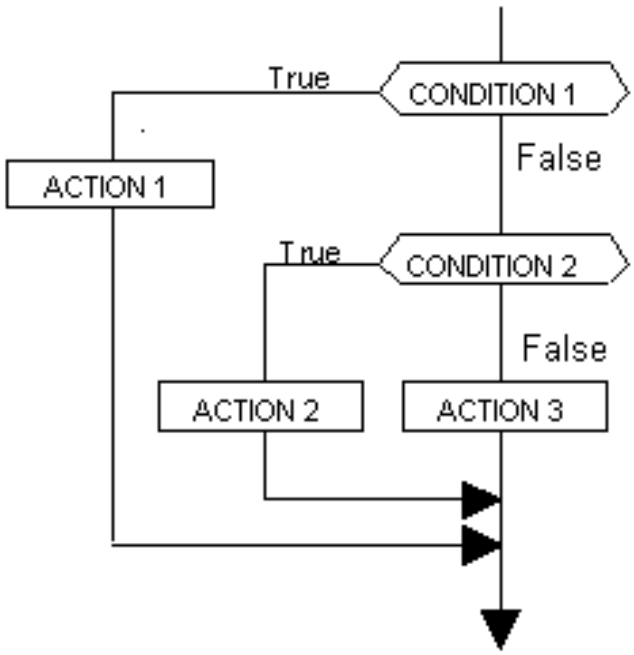


Рисунок 6.3 - Основна форма умовної дії IF... END_IF;

Приклад простої форми умовної дії IF... END_IF; :

```
IF %M0 AND %M1 THEN
      %M5:=%M3+%M4;
      SET %M10;
      ELSIF %M0 OR %M1 THEN
          %MW5:=%MW3-%MW4;
          SET%MW11;
      ELSE
          RESET %M10;
          RESET %M11;
END_IF;
```

Умовна повторювана дія WHILE ... END_WHILE;

Команда виконує періодично повторювану дію, поки умова виконується.

Приклад умовної повторюваної дії WHILE ... END_WHILE; :

```
WHILE %MW4<12 DO
      INC %MW4;
      SET %MW25[%MW4];
END_WHILE;
```

• Умова може бути множинною.

• Дія представляє собою список команд.

• Умова випробовується перед виконанням дії. Дія виконується, поки умова істинна.

• Можуть бути вкладеними кілька структур керування WHILE.

Syntax **Operation**

WHILE condition **DO**

action;

END_WHILE;

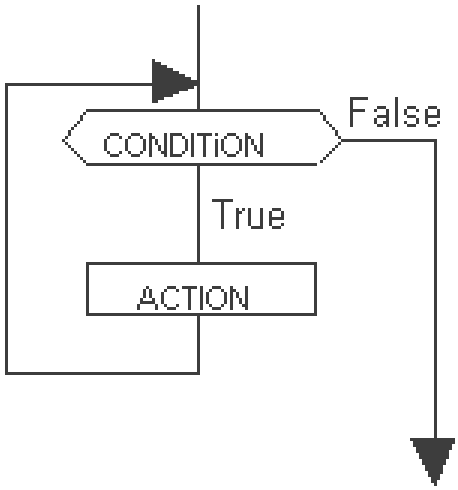


Рисунок 6.4 - Умовна повторювана дія WHILE ... END_WHILE;

Умовна дія, що триває REPEAT…END_REPEAT;
Команда виконує повторювану дію до перевіреної умови.

Syntax **Operation**

REPEAT

action;

UNTIL condition **END_REPEAT**;

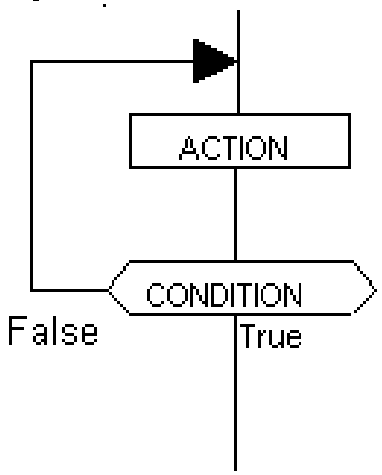


Рисунок 6.5 - Умовна дія, що триває REPEAT…END_REPEAT;

Приклад:

```
REPEAT
    INC %MW4.
    SET %MW25[%MW4];
UNTIL %MW4>12 END_REPEAT;
```

- Умова може бути множинною.
- Дія представляє собою список команд.
- Умова тестується, як тільки дія була виконана. Якщо умова не виконується, дія буде виконана ще раз.
- Можуть бути вкладеними кілька структур керування REPEAT.

Повторювана дія FOR ... END_FOR;

Команда виконує операцію з обробки кілька разів, збільшуючи індекс на 1 в кожному циклі.

• Коли індекс більший ніж остаточне значення, виконання продовжується на наступній команді за ключовим словом END_FOR.

• Індекс збільшується автоматично.

• Дія представляє собою список команд.

• Початкове значення і остаточне значення повинні бути чисельним виразом типу слова.

• Індекс повинен бути об'єктом типу слова, який є доступним у режимі читання.

• Можуть бути вкладеними кілька структур керування FOR.

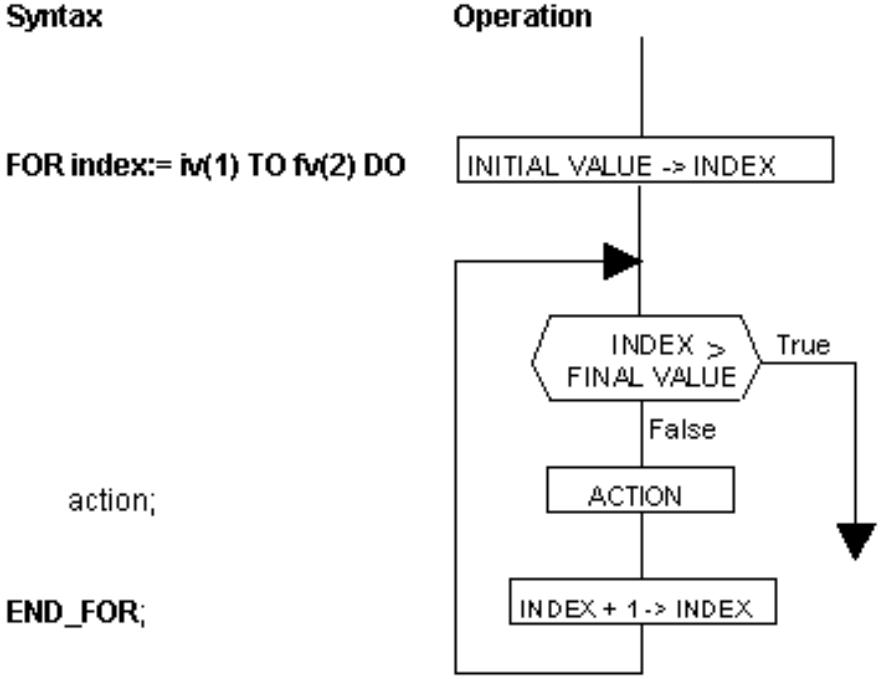


Рисунок 6.6 - Повторювана дія FOR ... END_FOR;

Приклад:

```
FOR %MW4:=0 TO %MW23+12 DO
    SET %M25[%MW4];
END_FOR;
```

Команда виходу EXIT

• Ключове слово EXIT використовується для виконання зупинки циклу, і продовження роботи на команді, яка йде за циклом.

• Команда може використовуватися тільки в одній із трьох дій: WHILE, REPEAT чи циклів FOR.

• Команда призначена до найближчого вкладеного циклу і не зупиняє виконання всіх циклів, що оточують її.

Приклад:

```
WHILE %MW1<124 DO
    %MW2:=0;
    %MW3:=%MW100[%MW1];
    REPEAT
        %MW500[%MW2]:=%MW3+%MW500[%MW2];
        IF(%MW500[%MW2]>32700) THEN
            EXIT;
        END_IF;
        INC %MW2;
    UNTIL %MW2<25 END_REPEAT;
    INC %MW1;
END_WHILE;
```

У цьому прикладі слово EXIT використовується для зупинки циклу REPEAT, але не циклу WHILE.

6.4 Правила для виконання програм Structured Text

Програма на мові Structured Text виконується послідовно, команда за командою з врахуванням структур керування.

У випадку арифметичних чи логічних виразів, що складаються з кількох операторів, визначаються правила пріоритету між різними операторами (таблиця 6.1).

Таблиця 6.1 - Пріоритет виразів на мові ST

Оператор	Символ	Пріоритет
Круглі дужки	(вираз)	Більш високий
Логічне доповнення Інверсія - на операнді + на операнді	NOT NOT - +	
Множення Ділення Модуль	* / REM	
Додавання Віднімання	+ -	
Порівняння	<, >, <=, >=	
Порівняння рівних Порівняння нерівних	= <>	
Логічний AND	AND	
Логічне ексклюзивне OR	XOR	
Логічне OR	OR	Більш низький

Приклад:

```
NOT %MW3 * 25 AND %MW10 + %MW12
```

У цьому прикладі, NOT виконується на %MW3, тоді результат множиться на 25. Розраховується сума %MW10 і %MW12, тоді виконується логічне AND між результатом множення і додавання.

Коли є конфлікт між двома операторами того ж самого пріоритету, перший оператор буде мати вищий пріоритет (оцінка виконується зліва направо).

Приклад:

```
%MW34 * 2 REM 6
```

У цьому прикладі %MW34 спочатку збільшується на 2, тоді результат використовується для знаходження модуля.

Використання круглих дужок

Круглі дужки використовуються для зміни порядку, у якому оператори оцінюються, наприклад, давати додаванню більш високий пріоритет, ніж множенню.

Приклад:

```
(%MW10 + %MW11) * %MW12
```

У цьому прикладі, додавання буде виконуватися перед множенням.

Круглі дужки можуть бути вкладеними; немає ніякого обмеження рівням вкладення. Круглі дужки можуть також використовуватися для запобігання неправильного тлумачення програми.

Приклад:

```
NOT %MW2 <> %MW4 + %MW6
```

Використовуючи пріоритетні правила, можна записати:

```
((NOT %MW2) <> (%MW4 + %MW6))
```

Користувач міг би так пробувати виконувати наступну операцію:

```
NOT (%MW2 <> (%MW4 + %MW6))
```

Цей приклад показує, як круглі дужки можуть використовуватися для прояснення програми.

Неявні перетворення

Неявні перетворення відносяться до слів і подвійних слів. Оператори, що використовуються в арифметичних виразах і виразах порівняння і оператори присвоєння виконують ці неявні перетворення.

Для команди форми: **<операнд 1> <оператор> < операнд 2>**, можливі такі перетворення:

Тип операнда 1:	Тип операнда 2:	Перетворення операнда 1	Перетворення операнда 2	Тип операції:
Слово	Слово	Немає	Немає	Слово
Слово	Подвійне слово	Подвійне слово	Немає	Подвійне слово
Подвійне слово	Слово	Немає	Подвійне слово	Подвійне слово
Подвійне слово	Подвійне слово	Немає	Немає	Подвійне слово

Для призначення форми **<лівий операнд> := <правий операнд>**, лівий операнд надає правому операнду очікуваний тип операнда, щоб виконати операцію. Це означає, що правий операнд повинен бути перетворений у разі потреби, відповідно до таблиці:

Тип лівого операнда	Тип правого операнда	Конверсія правого операнда
Слово	Слово	Немає
Слово	Подвійне слово	Слово
Подвійне слово	Слово	Подвійне слово
Подвійне слово	Подвійне слово	Немає

Список питань для самоконтролю

1. Які основні поняття мови Structured Text?
2. Які використовуються розрядні команди в Structured Text?
3. Які використовуються команди чисельного порівняння в Structured Text?
4. Які використовуються команди для операцій з розрядовими таблицями в Structured Text?
5. Які використовуються команди арифметичних дій в Structured Text?
6. Які використовуються логічні команди в Structured Text?
7. Які використовуються команди перетворення чисел в Structured Text?
8. Які використовуються команди для таблиць в Structured Text?
9. Які використовуються команди для символьних рідків в Structured Text?
10. Які використовуються програмні команди в Structured Text?
11. Які використовуються команди адміністрації часу в Structured Text?
12. Які основні принципи програмування в Structured Text?
13. Як використовуються коментарі в Structured Text?
14. Як використовуються мітки в Structured Text?
15. Які використовуються структури керування в Structured Text?
16. Який приорітет мають різні вирази в Structured Text?
17. Як використовуються неявні перетворення в Structured Text?

7 МОВА ПРОГРАМУВАННЯ GRAFCET

7.1 Історія розвитку

У 1975 році в робочій групі “Логічні системи” організації AFCET (Association francaise de cybernetique economique et technique - Французька асоціація економічної і технічної кібернетики) було прийняте рішення про створення комісії “Нормалізація представлення технічного завдання системи автоматизації”. У комісію увійшли провідні французькі університети і виробники. Мета комісії: створення формального представлення поведінки об'єкта і його керування, яке буде просте і прийнятне для всіх учасників проекту - від розробника системи керування до експлуатаційного персоналу – яке може бути легко реалізоване у відповідному аппаратно-програмному комплексі автоматизації. Як вихідні інструменти були розглянуті три класи засобів моделювання: органіграми, мережі Петрі і графи станів. Аналіз цих і інших методів у 1977 році привів до створення моделі, названої **GRAFCET** (grAFCET, gr - «граф»). З тих пір Grafcet пройшов 4 основні етапи розвитку:

1-й етап: 1975 -1977 народження Grafcet.

2-ий етап: 1978-1982. Grafcet успішно поширюється в сфері освіти і промисловості. У 1982 році він стає французьким стандартом (NFC03-190). Перші реалізації Grafcet у ПЛК зроблені фірмами Merlin Gerin і Telemecanique.

3-й етап: 1983-1988. GRAFCET включається в програму навчання середніх і вищих технічних навчальних закладів. У промисловості він прийнятий усіма французькими виробниками ПЛК, росте його популярність у Європі й у світі. У 1988 році він стає міжнародним стандартом для програмування ПЛК IEC848 і в англомовній термінології одержує назву **SFC - Sequential Function Chart**. Величезна роль в інтернаціоналізації GRAFCET належить фірмі Telemecanique - лідеру французького ринку засобів автоматизації і промислового контролю.

4-й етап: 1989-1996. Група “Логічні системи” одержує нову назву “Група Grafcet” і підсилює проробку теоретичних аспектів. Дійсно, зростання вимог користувачів АСУ, що йдуть у свою чергу від зростання складності автоматизованих процесів і машин вимагають додаткового розвитку моделі і реалізації Grafcet, зокрема, розширених можливостей надійного функціонування і діагностики, відкритості до інших інструментів моделювання та ін. Розробки групи стають предметом обговорення на багатьох конференціях і конгресах, дискусії посилюються після відкриття сервера Grafcet у Internet.

Паралельно, у 1990 була створена підгрупа “Grafcet і навчання”. Її задачами були розвиток і просування Grafcet у сфері середньої і вищої технічної освіти.

У 1993 р. Grafcet за назвою SFC входить у стандарт IEC 1131.

7.2 Основні принципи Grafcet

Grafcet використовується для представлення послідовних операцій системи керування графічним і структурним способами. Це графічне описання послідовних операцій системи керування і різних ситуацій, що відбуваються у системі, виконаних з використанням простих графічних символів (рисунок 7.1).

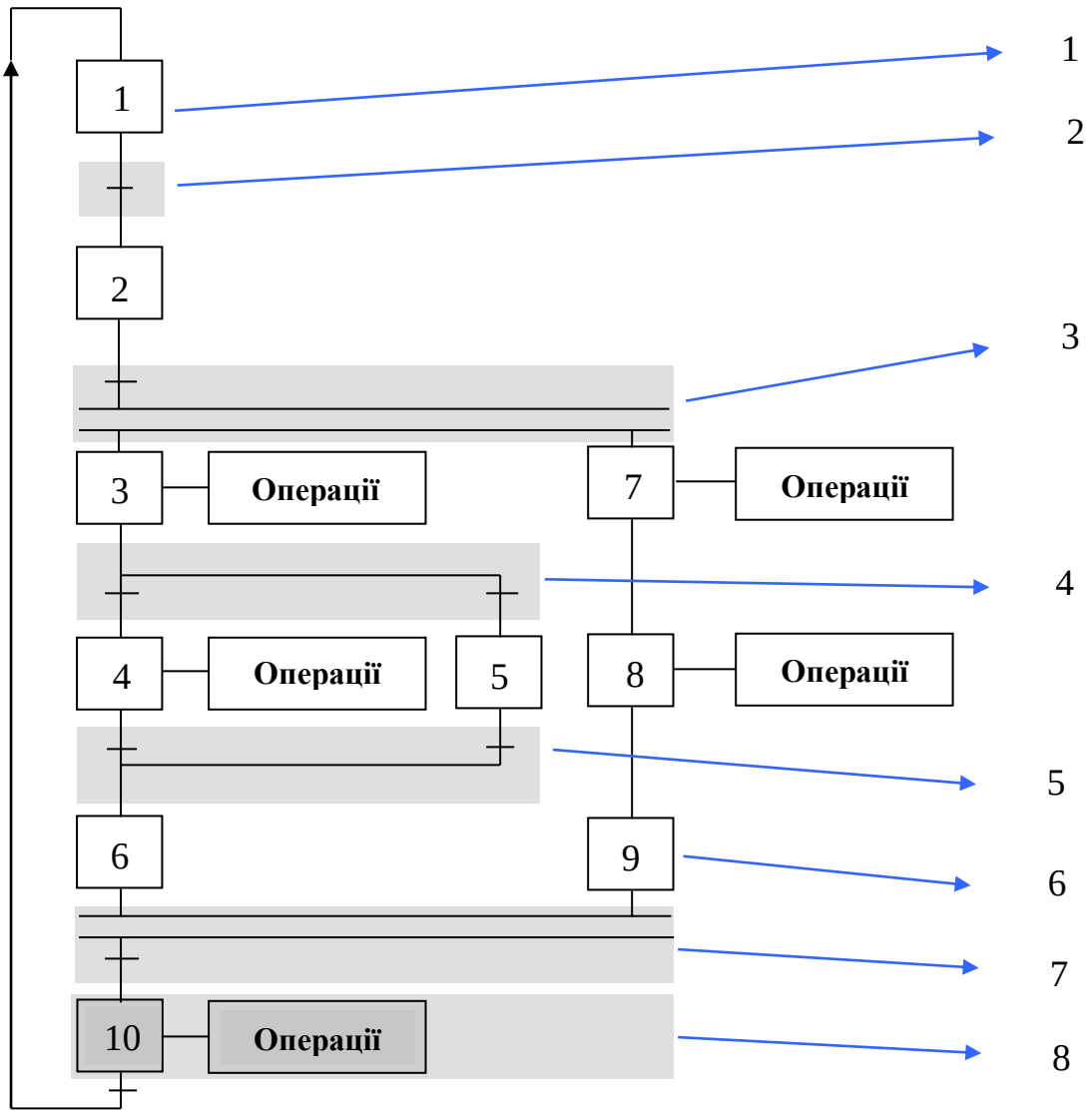


Рисунок 7.1 – Основні кроки мови Grafcet

1 - Початковий крок: визначає початкову ситуацію PLC.

2 - Перехід: зв'язані **умови переходу** вказують логічні умови, необхідні для очищення цього переходу.

3 - Одночасна активація кроків 3 і 7 (AND дивергенція). Ряд кроків 3, 4, 5, 6 і 7, 8, 9 складають дві паралельні послідовності.

4 - Вибір послідовності (OR дивергенція) від кроку 3 до кроку 4 чи кроку 5.

5 - Кінець вибору послідовності (OR ковергенція) від кроку 4 чи кроку 5 до кроку 6.

6 - Кінець послідовності: дозволяє синхронізацію послідовностей.

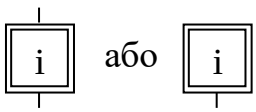
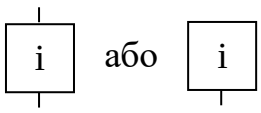
7 - Одночасна деактивація кроків 6 і 9 (AND ковергенція).

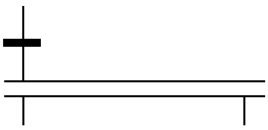
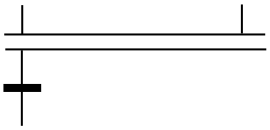
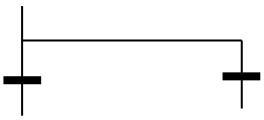
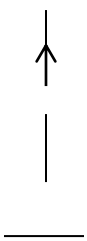
8 - Крок: відповідні дії виконуються тільки тоді, коли цей крок активний.

Переходи і прямі зв'язки представлені в символічній формі можливості прогресії **активних кроків**.

Дії, пов'язані з кроками, вказують на основні умови “що повинно бути виконано”, коли вони активні. Зокрема вони описують команди, які повинні бути послані діючій частині (процес автоматизації) чи іншим автоматизованим системам. Набір активних кроків у будь-який даний час визначає поточне місце виконання програми Grafcet.

7.3 Графічні символи мови Grafcet

Назва	Символ	Функції
Початкові кроки		Вказує початкові кроки, активні на початку циклу після ініціалізації чи холодного рестарту
Одинарні кроки		Вказує, що система керування знаходиться в стійкому стані. Кількість кроків, які можуть бути сконфігуровані: - Від 1 до 96 для TSX 37-10. - Від 1 до 128 для TSX 37-20 чи TSX 57.

Переходи		Використовуються для переходу від одного кроку до іншого. Умова переходу, зв'язана з цим переходом використовується для визначення логічних станів, потрібних для очистки цих переходів. Кількість кроків, які можуть бути сконфігуровані: - Від 1 до 96 для TSX 37-10. - Від 1 до 128 для TSX 37-20 чи TSX 57.
AND дивергенції		Перехід від одного кроку до кількох кроків. Використовується для активації максимум 11 кроків одночасно.
AND ковергенції		Перехід від декількох кроків до одного кроку. Використовується для деактивації максимум 11 кроків одночасно.
OR дивергенції		Перехід від кількох кроків до одного кроку. Використовується для виконання вибору послідовності максимум 11 кроків.
OR ковергенції		Перехід від кількох кроків до одного кроку. Використовується як кінець вибору послідовності від максимум 11 кроків.
З'єднувач джерела		'n' - номер кроку, від якого прийшло керування (крок джерела).
З'єднувач призначення		'n' - номер кроку, в який йде керування (крок призначення).
Прямі зв'язки: Вверх Вниз Вліво або вправо		Ці зв'язки використовуються для вибору послідовності при переході на більше ніж один крок, при переході до повторюваних кроків.

7.4 Об'єкти Grafcet

Програміст має доступні йому біти об'єктів, які зв'язані з кроками, системні біти мови Grafcet, слово об'єктів, яке вказує активний час кроків і системні слова мови Grafcet.

Назва	Адреса	Опис
Біти кроку	%Xi	Стан кроку і головної схеми Grafcet (i=0...n) (n залежить від процесора)
Системні біти Grafcet	%S21	Ініціалізує схему Grafcet
	%S22	Повторна установка всіх схем Grafcet в 0
	%S23	Заморожування схеми Grafcet
	%S26	Установка в 1 на: - Переповнення таблиці (кроки/переміщення) - Виконання неправильної частини (з'єднувач призначення на крок, який не належить даній частині)
Слова кроку	%Xi.T	Активний час кроку і головної частини Grafcet
Системні слова Grafcet	%SW20	Слово індикації для поточного циклу: число активних кроків, що були активованими і деактивованими.
	%SW21	Слово індикації для поточного циклу: число правильних переходів, що були активованими чи деактивованими.

Біти кроку %Xi

• Біти у стані 1, коли кроки активні.

• Дані біти можна використати у всіх задачах, але можуть бути описані тільки при попередній обробці головної задачі (ініціальзація програми). Дані тести і дії запрограмовані мовами Ladder Diagram, Instruction List чи Structured Text.

• Дані біти не можуть бути індексовані.

Слова активного часу кроків % Xi.T

• Збільшуються кожні 100 мс і мають значення від 0 до 9999.

• Інкрементація слова: протягом активності відповідного кроку.

• При деактивації кроку його вміст заморожується.

• При активації кроку його вміст перевстановлюється, тоді слово інкрементується.

• Кількість слів активного часу кроків не може бути сконфігуровано, одне слово резервується для кожного кроку.

• Дані слова не можуть бути індексовані.

7.5 Представлення частини Grafcet

Головна частина може бути запрограмована на 8 сторінках (сторінки від 0 до 7). Кожна сторінка Grafcet (рисунок 7.2) має 14 рядків і 11 стовпців, які визначають 154 клітинки. Один графічний елемент може бути введений у одну клітинку.

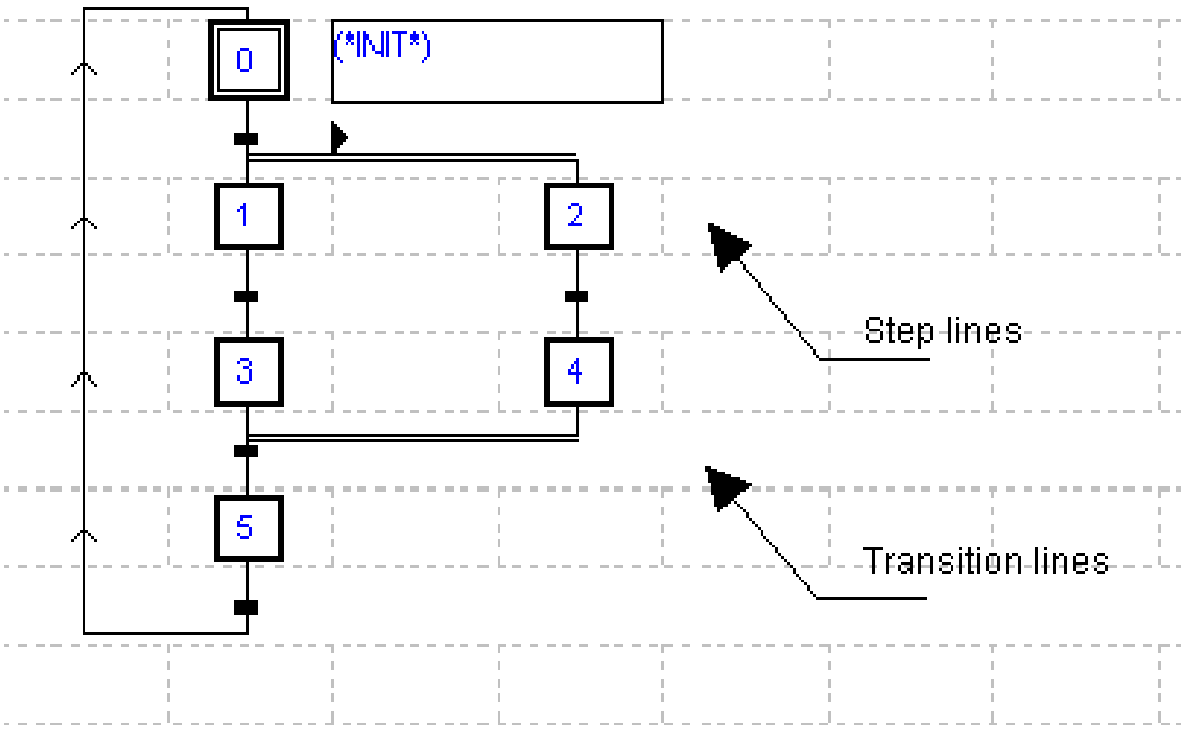


Рисунок 7.2 – Сторінка Grafcet

Правила запису

- Перший рядок використовується для введення з'єднувачів джерела.
- Останній рядок використовується для введення з'єднувачів призначення.
- Парні рядки (від 2 до 12) - рядки кроків (для кроків і з'єднувачів призначення).
- Непарні рядки (від 3 до 13) - рядки переходів (для переходів і з'єднувачів джерела).
- Кожен крок пронумерований (від 0 до 127) у будь-якому порядку.
- Кілька частин можуть бути представлені на одній сторінці.

Вибір послідовності і закінчення вибору послідовності

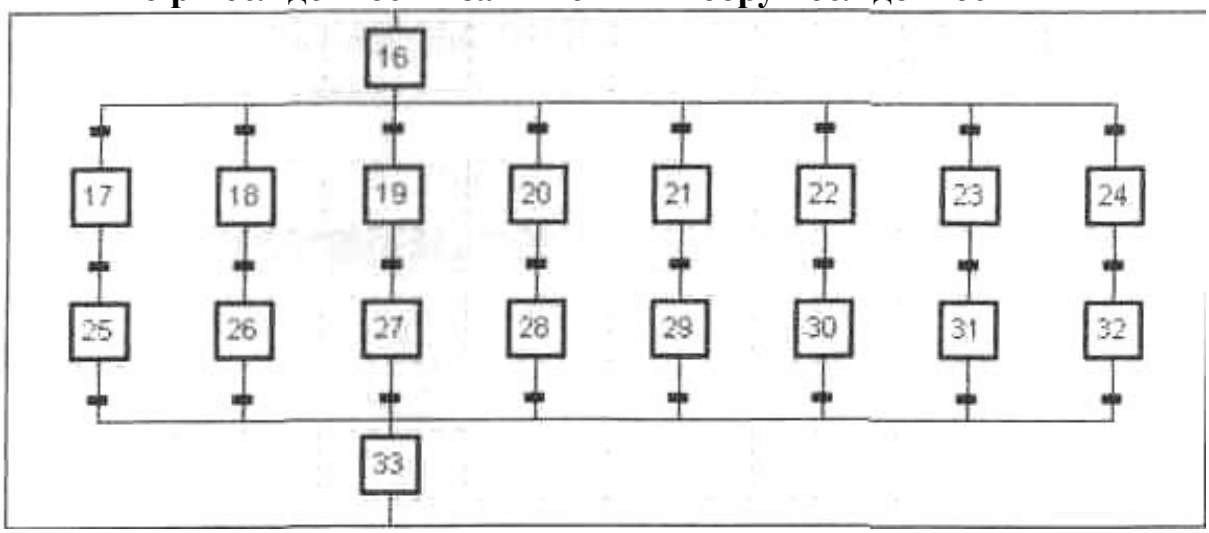


Рисунок 7.3 – Вибір послідовності в Grafcet

- Кількість OR ковергенцій чи OR дивергенцій не повинна перевищити 11.
- Вибір послідовності може бути спрямований ліворуч чи праворуч.
- Вибір послідовності повинен, взагалі, закінчитися кінцем вибору послідовності.
- Щоб уникнути очищення кількох переходів одночасно, відповідні умови переходу повинні бути ексклюзивними.

Одночасна активація і деактивація кроку

- Кількість кроків вниз одночасної активації (AND дивергенція) чи вниз одночасної дезактивації (AND ковергенція) не повинна перевищити 11.
- Одночасна активація кроків повинна, взагалі, закінчитися одночасною дезактивацією кроків.
- Одночасна активація завжди відбувається зліва направо.
- Одночасна деактивація завжди відбувається зправа наліво.

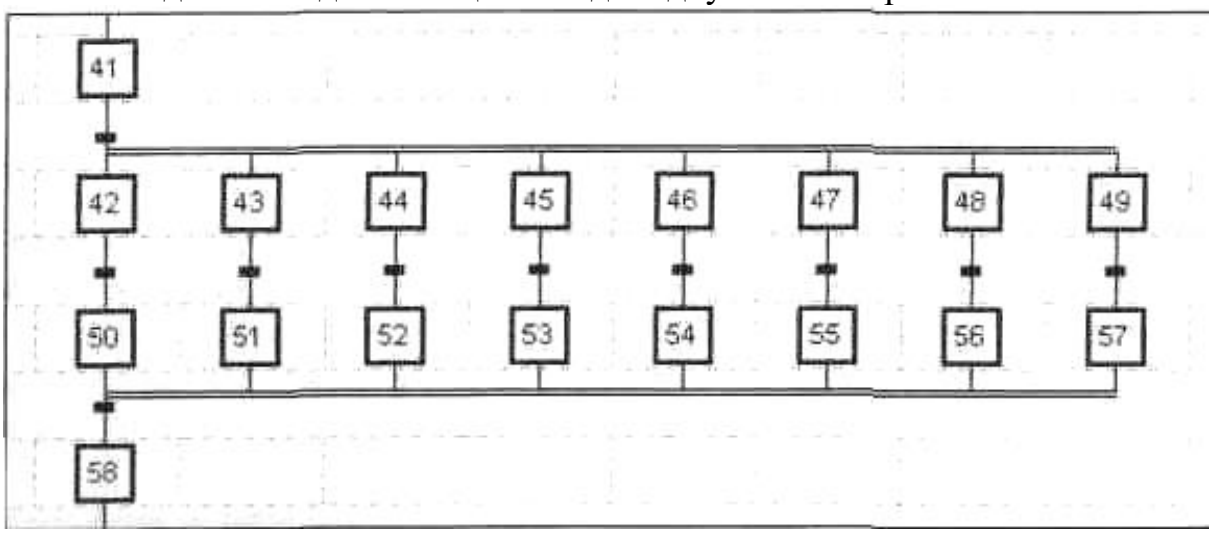


Рисунок 7.4 - Одночасна активація і деактивація

Використання з'єднувачів

Метою з'єднувачів є гарантія неперервності частини Grafcet, коли прямі зв'язки, чи на одній сторінці чи між двома послідовними сторінками, не можуть бути графічно представлені. Ця неперервність забезпечується з'єднувачем призначення, що завжди має відповідний з'єднувач джерела.

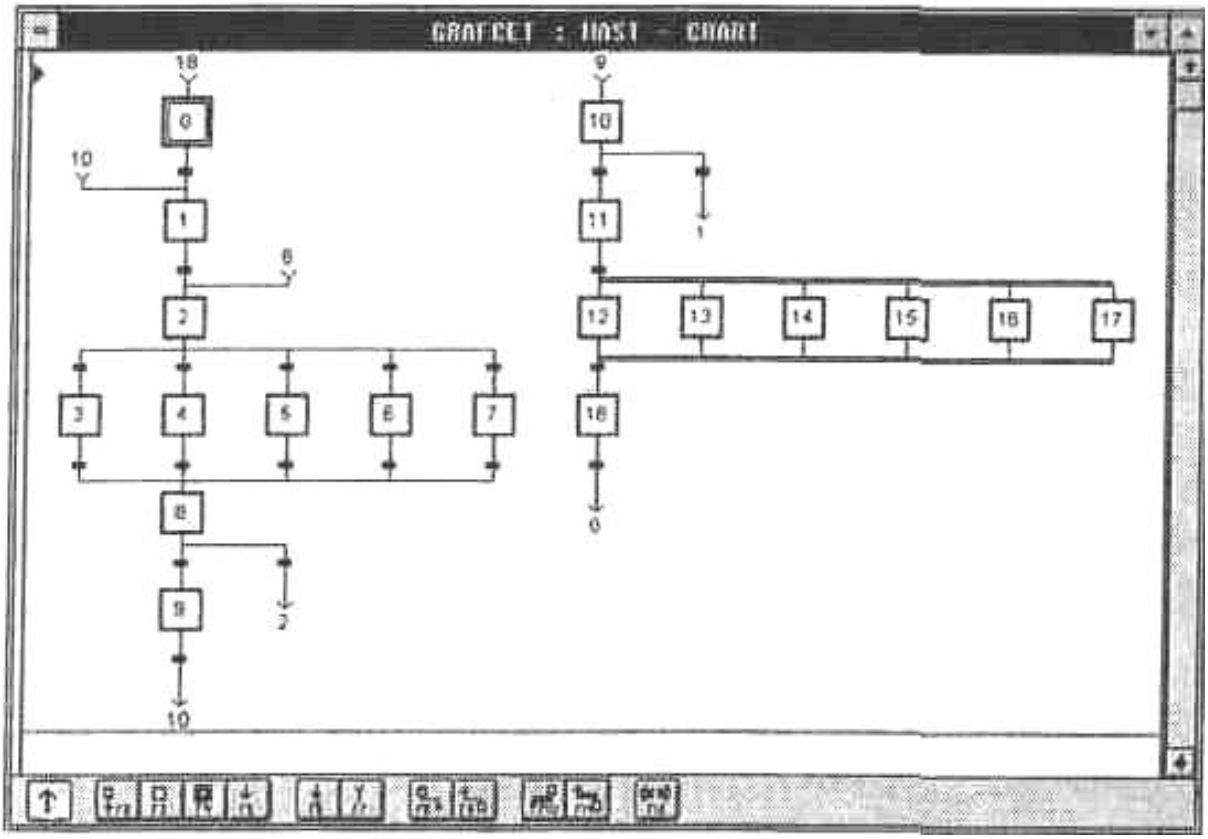


Рисунок 7.5 - Використання з'єднувачів

Частина може бути циклічною назад, використовуючи з'єднувачі (наприклад, цикл від кроку 18 до кроку 0).

Послідовність може бути перезапущена, використовуючи з'єднувачі (наприклад, крок 10 до кроку 1 чи крок 8 до кроку 2).

З'єднувачі використовуються, коли гілка частини довша, ніж сторінка (наприклад, крок 9 до кроку 10).

З'єднувачі для вибору послідовності і закінчення вибору послідовності

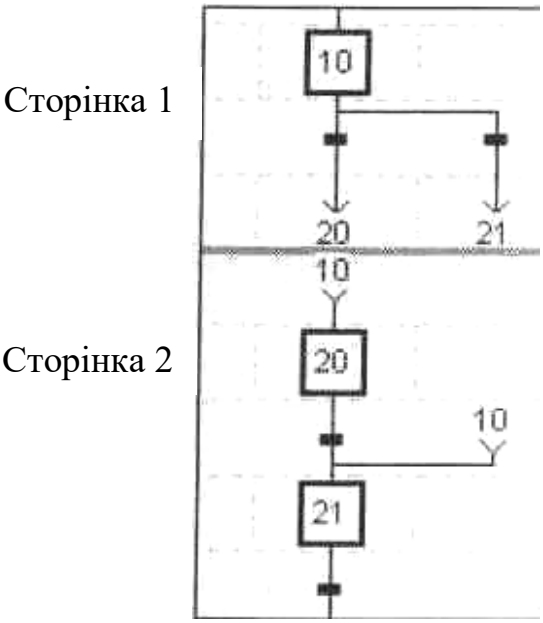


Рисунок 7.6 - З'єднувачі для вибору послідовності і закінчення вибору послідовності

- Для вибору послідовності, переходи і з'єднувачі призначення повинні бути введені в ту ж саму сторінку.

- Для закінчення вибору послідовності з'єднувачі джерела повинні бути введені в ту ж саму сторінку, що і крок призначення.

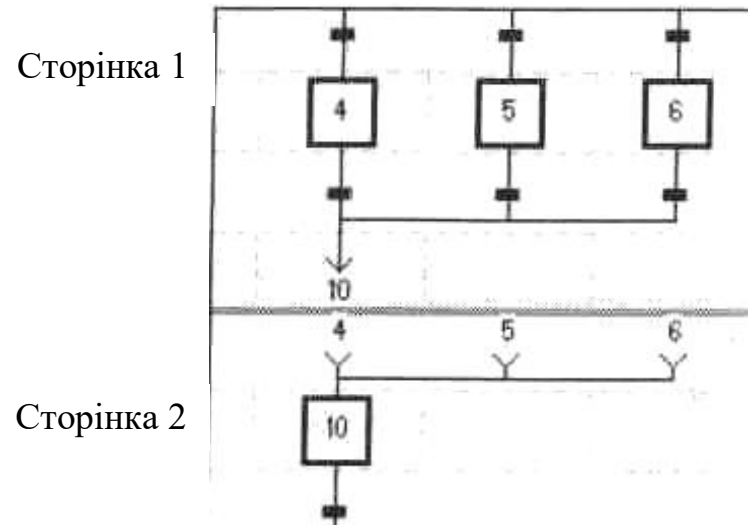
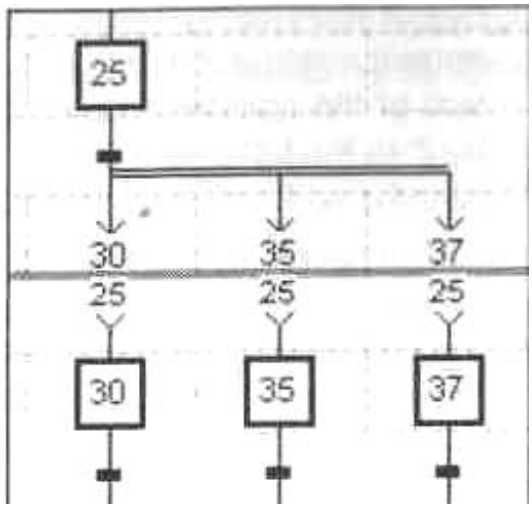


Рисунок 7.7 - Кінець вибору послідовності

- Кінець вибору послідовності здійснюється з'єднувачем призначення, він повинен мати таке ж саме число з'єднувачів джерела, як кроки перед закінченням вибору послідовності.

З'єднувачі для одночасної активації і деактивації кроків

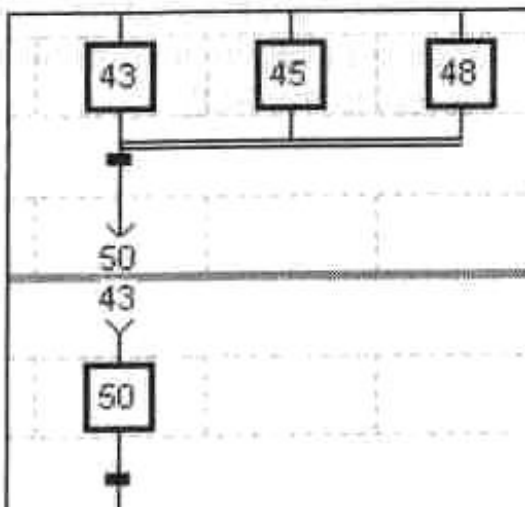


Сторінка 2

Сторінка 3

- Для одночасної активації кроків, з'єднувачі призначення повинні бути на тій же самій сторінці, що й крок та дівергенція переходу.

Рисунок 7.8 – Одночасна активація кроків



Сторінка 4

Сторінка 5

- Для одночасної деактивації кроки і конвергенція переходу повинні бути на тій же самій сторінці, що і з'єднувач призначення.

Коли кілька кроків сходяться на одному переході, з'єднувач джерела має номер верхнього кроку, розташованого лівіше всіх інших кроків.

Рисунок 7.9 – Одночасна деактивація кроків

Прямі зв'язки

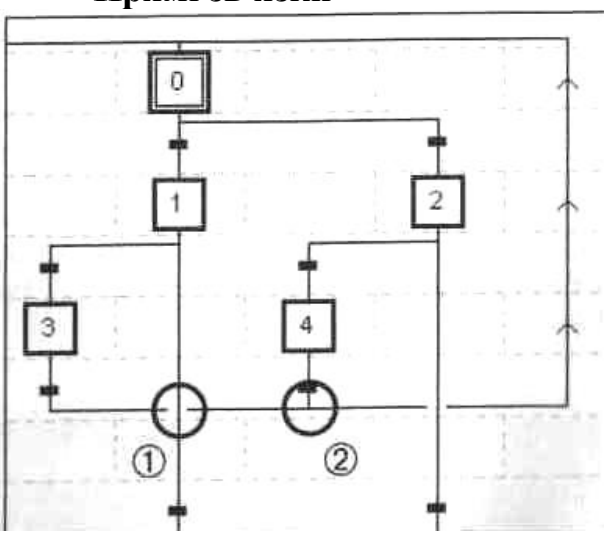


Рисунок 7.10 – Прямі зв'язки

· Прямі зв'язки з'єднують крок з переходом чи перехід із кроком. Вони можуть бути вертикальними або горизонтальними

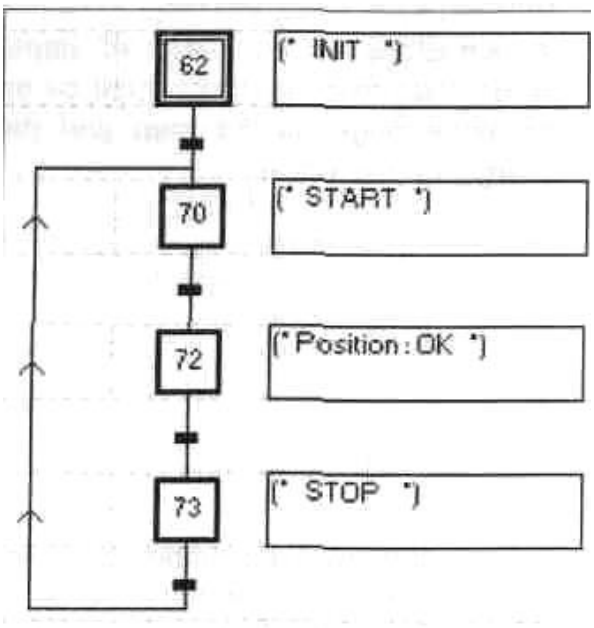
· Прямі зв'язки можуть:

- перетинатися ① , будучи різних типів

- зустрічатися ② , будучи того ж самого типу.

· Зв'язок не може перетинати одночасно крок активації чи деактивації.

Коментарі



- У сторінку Grafcet можна ввести коментарі в будь-яку клітину. Текст коментаря обмежується символами (* і *). Максимальний розмір - 64 знаки.
- Коментар займає дві суміжні клітинки максимум на двох рядках. Якщо зона дисплея занадто мала, коментарі коротшають, щоб відповідати дисплею, але при друкуванні документа, коментар показується повністю.
- Коментар, введений у сторінку Grafcet, зберігається у графічних даних, завантажених в PLC.

Рисунок 7.11 – Коментарі

7.6 Дії, які пов'язані з кроками

Кожен крок має відповідні дії, які можуть бути запрограмовані в Ladder Diagram, Instruction List чи Structured Text. Ці дії скануються, якщо крок, з яким вони пов'язані, активний. Програмне забезпечення PL7 дозволяє три типи дій:

- **Дії на активацію**: дії, виконані один раз, при активації відповідного їм крока.
- **Дії на дезактивацію**: дії, виконані один раз, при деактивації відповідного їм крока.
- **Неперервні дії**: дії, виконані неперервно так довго, як крок, з яким вони пов'язані.

Ці три типи дій можуть використовуватися для кожного кроку.

Одинарна дія може містити кілька елементів програмування (послідовності, твердження, ступені).

Дії звертання

Ці дії формулюються таким чином:

MAST - CHART - PAGE n %Xi x

де: x = P1 → Активація
= N1 → Неперервний
= P0 → Деактивація

n = Номер сторінки
i = Номер кроку

Приклад: MAST - CHART - PAGE 0 %X1 P1
Дія на активацію крока 1 сторінки 0

Правила використання

- Усі дії розглядаються як дії зберігання, отже:
  - дія, що керує тривалістю кроку Xn, повинна бути повторно встановлена на деактивації кроку Xn чи активації кроку Xn+1;
  - дія, що впливає на кілька кроків, встановлена в 1 на активації кроку Xn і очищається на деактивації кроку Xn+m.
- Усі дії можуть керуватися логічними умовами, тобто бути умовними.

Дії на активацію чи деактивацію

Дані дії імпульсні і виконуються на одному скануванні. Вони використовуються для виклику підпрограми, інкрементують лічильник і т.д.

Приклад 1: Виклик підпрограми (рисунок 7.12)

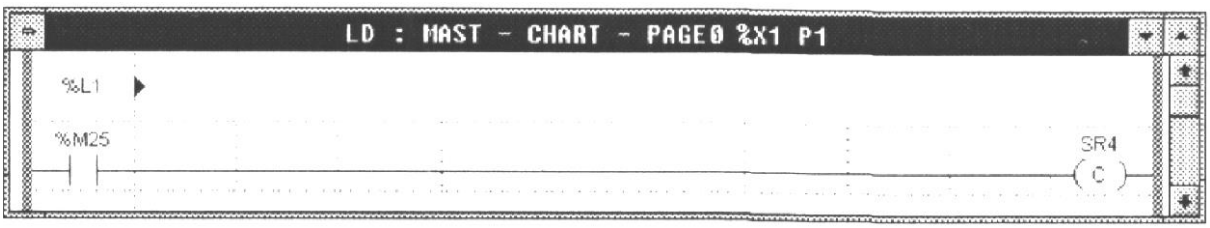


Рисунок 7.12 – Ілюстрація до прикладу 1

Приклад 2: Інкремент слова %MW10 і скидання значень %MW0 та %MW25 (рисунок 7.13)

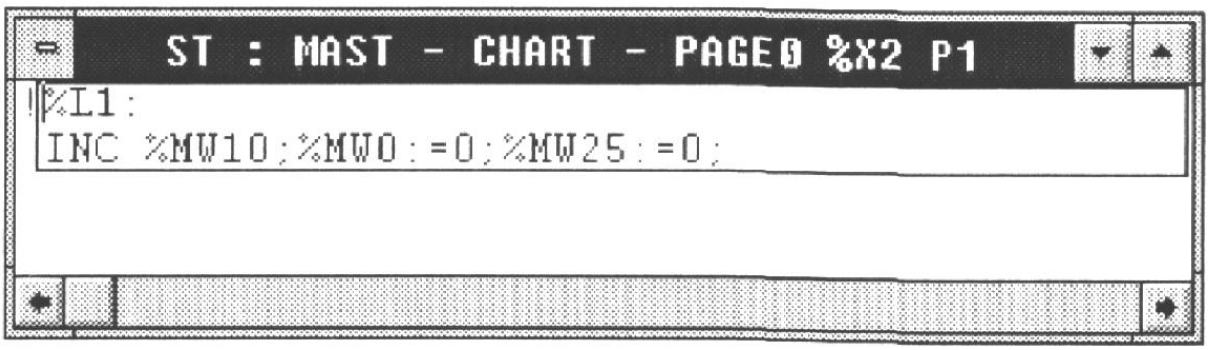


Рисунок 7.14 – Ілюстрація до прикладу 2

Неперервні дії

• Умовна дія

Приклад:

Біт %M10 керується входом %I2.5 чи внутрішнім бітом %M9 і входом %I1.2.

Поки крок 2 активний і ці умови присутні, %M10 встановлюється в 1. Останній стан читається на деактивації і записаний в пам'ять, тому що відповідні дії скануються не довго.

Це необхідно для повторної установки біта %M10 в 0, у дії на деактивації кроку для даного прикладу.

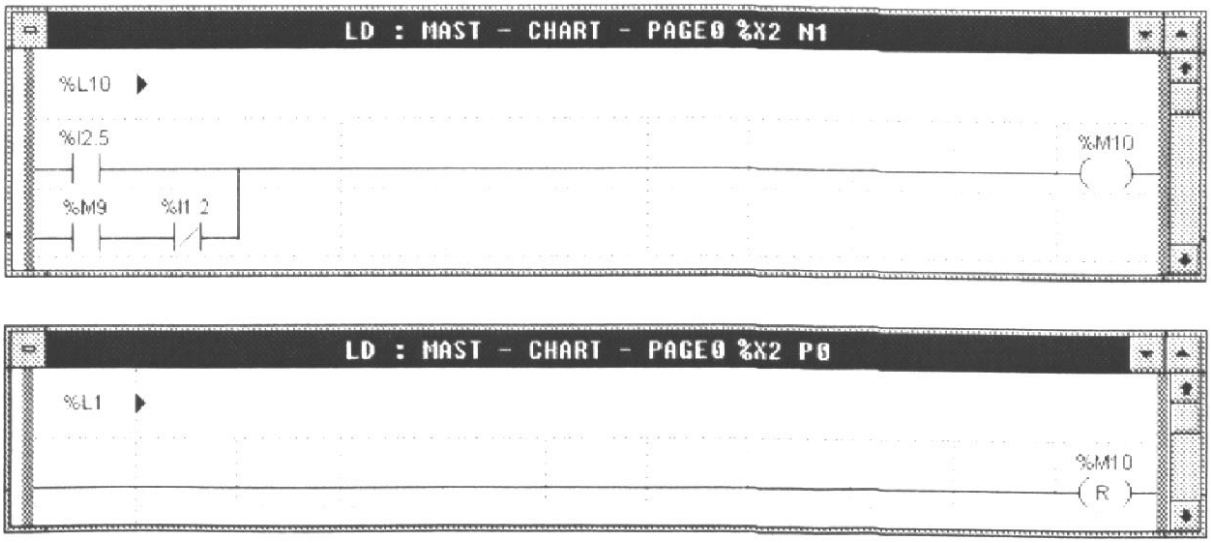


Рисунок 7.15 – Ілюстрація до прикладу

• Тимчасова умовна дія

Це окремий випадок, у якому час є логічною умовою. Цей зв'язок може бути виконано просто, тестуючи активний час, зв'язаний із кроком.

Приклад: Біт %M12 керується поки активний час кроку 3 менший ніж 10 секунд (базовий час: 100 мс).

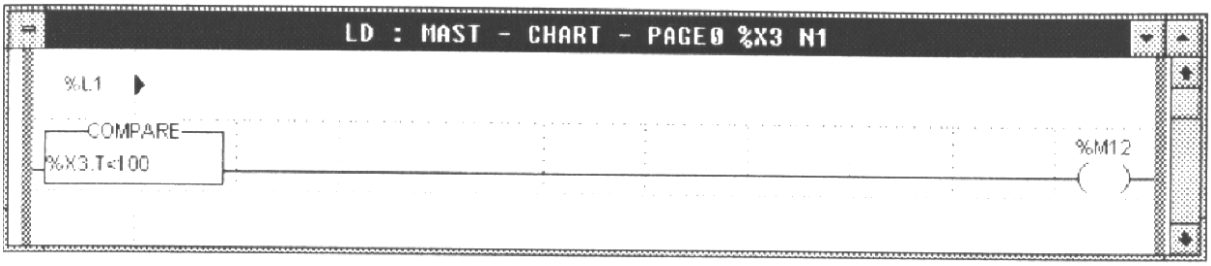


Рисунок 7.16 – Ілюстрація до прикладу

Ці дії можуть також бути не обмежені умовами.

Порядок виконання дій

У наступному прикладі, на одне сканування, порядок виконання дій такий:

Коли крок 51 активний, дії виконані в такому порядку:

1. Дії на дезактивацію кроку 50,
2. Дії на активацію кроку 51,
3. Неперервні дії кроку 51.

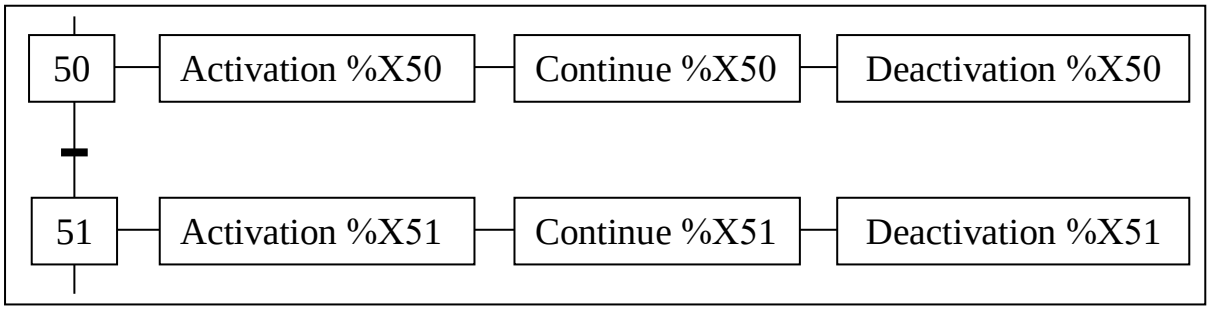


Рисунок 7.17 – Порядок виконання дій

Коли крок 51 деактивований, відповідні йому неперервні дії скануються недовго.

7.7 Умови, пов'язані з переходами

✓ Кожен перехід має пов'язану з ним умову, яка може бути запрограмована в Ladder Diagram, Instruction List чи Structured Text.

✓ Умова переходу сканується, коли перехід, з яким вона пов'язана, активний.

✓ Умова переходу відповідає блоку, списку команд чи твердженням Structured Text, включаючи ряд тестів на бітах і/чи словах.

✓ **Умова переходу, що не запрограмована - завжди помилкова умова переходу.**

Звертання на умови переходу

Умови переходу формулюються таким чином:

MAST - CHART - PAGE n %X(i) → %X(j)

де: n – Номер сторінки
i – Номер кроку вперед
j – Номер кроку назад

Приклад: MAST - CHART - PAGE 0 %X(0) → %X(1)

Умова переходу, пов'язана з кроком 0 і кроком 1 нульової сторінки карти.

Правила для програмування мовою Ladder Diagram

Умова, пов'язана з переходом, програмується у формі кроку, що включає зону тестування і зону дії.

Структура кроку така ж сама як кроку, запрограмованого в програмному модулі.

Можуть бути використані тільки такі елементи:

- Графічні тестові елементи: контакти (%Mi, %I, %Q, %TMi.D, тощо), блоки порівняння;
- Графічні елементи дій: тільки спіраль умови переходу (інші спіралі не істотні в цьому випадку).

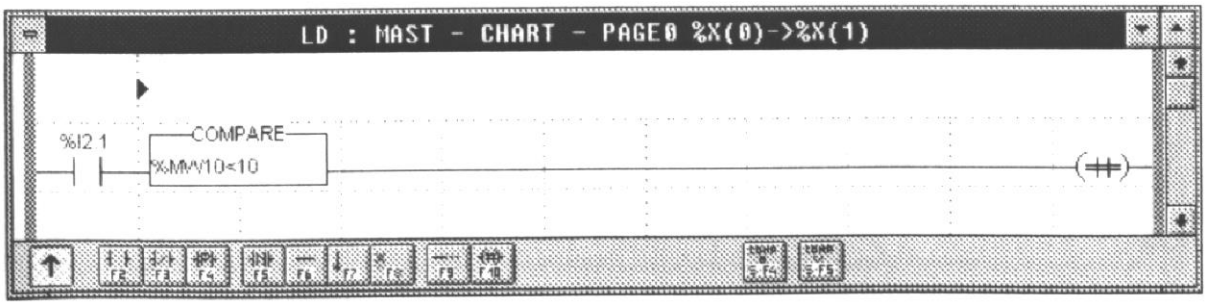


Рисунок 7.18 - Програмування умов мовою Ladder Diagram

Правила для програмування мовою Instruction List

Умова переходу програмується у формі списку команд, що містять тільки тестові команди.

Список команд для запису умови переходу відрізняється від стандартного списку такий чином:

- Основна структура: немає мітки (%L).
- Список команд:
  - немає команд дій (бітів об'єктів, слів чи функціональних блоків);
  - немає переходів, викликів підпрограм.

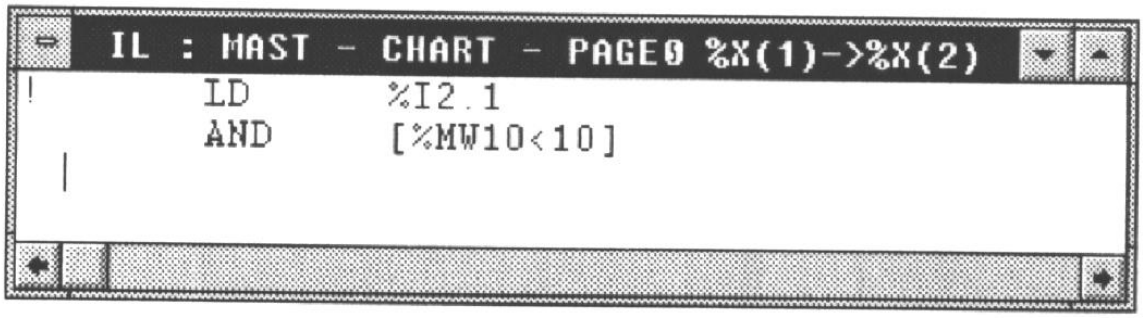


Рисунок 7.19 - Програмування умов мовою Instruction List

Правила для програмування мовою Structured Text

Умова переходу програмується у формі логічного виразу, чи арифметичного виразу, чи комбінації з них двох.

Вирази для запису умови переходу відрізняються від програмування мовою Structured Text:

- Основна структура:
  - немає мітки (%L),
  - немає тверджень дій, умовних тверджень чи повторюваних тверджень.
- Список команд:
  - Немає дій на біти об'єктів,
  - Немає переходів, виклику підпрограм,
  - немає переносів, немає команди дії на блоки.

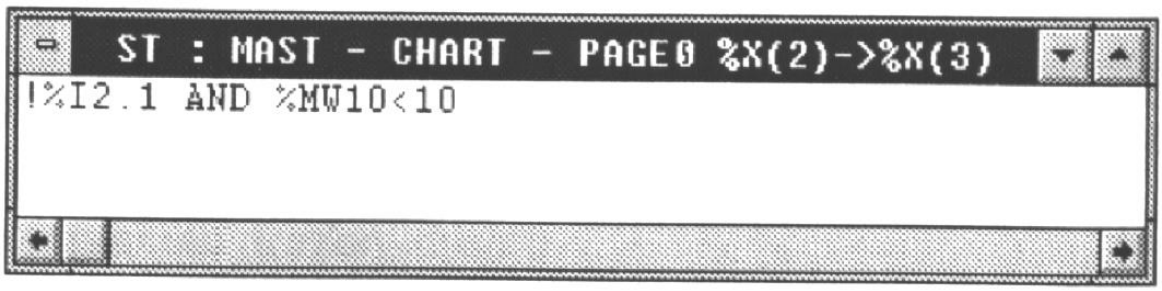


Рисунок 7.20 - Програмування умов мовою Structured Text

Умова переходу, що використовують активний час кроку

У деяких використаннях, дії керуються без контролю даних зворотного зв’язку (кінець переміщення, детектор, тощо). Тривалість кроку зумовлена часом: програмне забезпечення PL7 дає можливість активний час зв'язуватись з кожним кроком.

Приклад: Якщо користувач бажає залишитися в кроці 3 на час 15 секунд, умова для переходу між кроком 3 і кроком 4 буде (наприклад, мовою Structured Text):

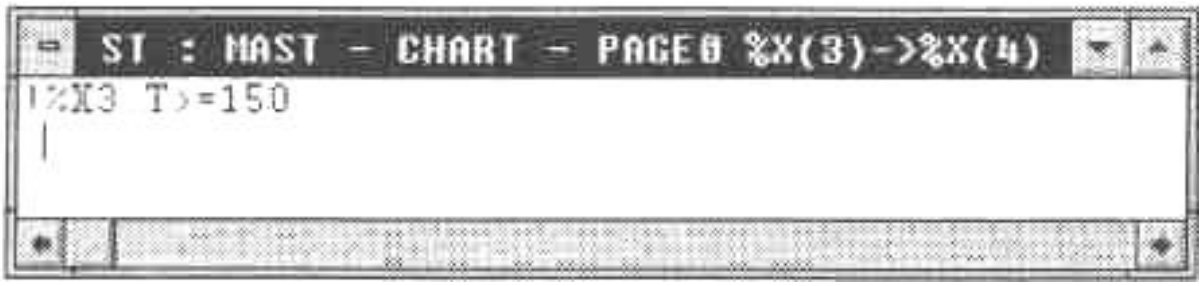
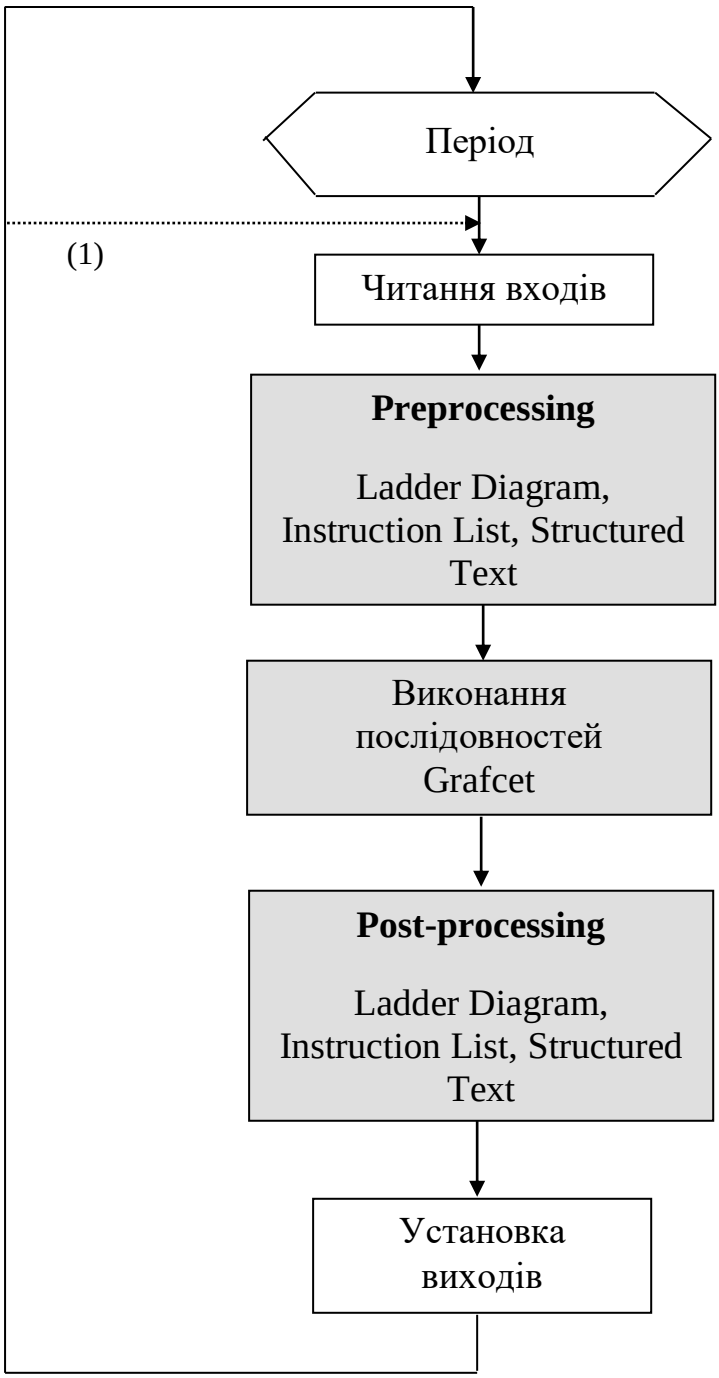


Рисунок 7.21 – Приклад умови переходу, яка використовує активний час кроку

7.8 Організація головної задачі

7.8.1 Опис головної задачі

Програма, написана на мові Grafcet, має три послідовних етапи роботи: попередні обчислення (preprocessing), виконання послідовностей Grafcet і кінцеві обчислення (post-processing) (рисунок 7.22).



(1) – при циклічній задачі

Рисунок 7.22 – Алгоритм роботи головної задачі

Період (при періодичній задачі) – це час між двома скануваннями задачі, який визначається конфігурацією.

При циклічній задачі входи читаються після модифікації виходів.

Читання входів:

Читання фізичних станів модулів введення PLC (значення, заморожені протягом обробки задачі).

Попередні обчислення використовуються при:

- ініціалізації програми при збої чи поверненні живлення,
- попередньому настроюванні частини Grafcet,
- обробці логічних входів.

Послідовна обробка використовується при обробці послідовної структури задачі.

Кінцеві обчислення використовуються при обробці:

- логічних виходів,
- контролю і захисних блокувань, які призначені для виходів.

Модифікування виходів – це модифікування фізичного стану модулів виходів PLC.

7.8.2 Попередні обчислення (preprocessing)

Програма попередньої обробки пишеться на мовах Ladder Diagram, Instruction List чи Structured Text і сканується повністю зверху вниз.

Вона виконується перед виконанням основних частин Grafcet для обробки всіх подій, які впливають на них:

- керування поверненням живлення і переініціалізації,
- скидування чи передустановка частин Grafcet.

Попереднє настроювання частини Grafcet

При програмуванні може бути необхідне попереднє настроювання частини Grafcet при зміні від нормального функціонування до визначеного режиму чи роботи в аварійних режимах (приклад: несправність, що викликає погіршення технологічного процесу).

Ця операція впливає на нормальне функціонування сканування програми і тому повинна використовуватися з обережністю. Попереднє настроювання може бути застосовано до всіх або до окремих частин Grafcet.

Приклад програми попереднього настроювання наведено на рисунку 7.23.

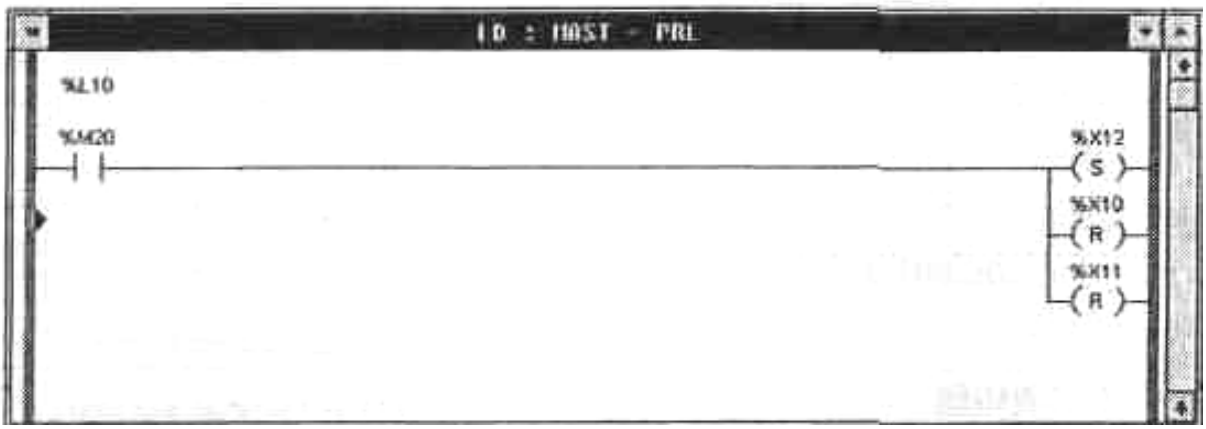


Рисунок 7.23 - Приклад програми попереднього настроювання

7.8.3 Послідовні обчислення частин Grafcet (sequential processing)

Цей розділ обчислень використовується для послідовної обробки структури Grafcet. Послідовна обробка складається з головної частини Grafcet, організованої з 8 сторінок.

В основній частині кілька незв'язаних частин Grafcet можуть бути запрограмовані і виконуватися одночасно.

Принцип розвитку

Розвиток частин Grafcet проводиться таким чином:

Стадія 1:

1. Оцінка стану переходів.
2. Запит на деактивацію відповідних верхніх кроків.
3. Запит на ініціювання нижніх кроків.

Стадія 2:

Розвиток стану частини Grafcet як функції переходів:

1. Деактивація верхніх кроків переходів.
2. Активація нижніх кроків переходів.
3. Повторний огляд правильності переходів.
4. Перевірка правильності нижніх переходів при активації нових кроків.

Система модифікує дві таблиці, які призначені для керування діяльністю кроків і переходів:

- **Таблиця активного кроку** для поточного сканування активних кроків, кроків, які будуть активовані і кроків, що будуть деактивовані,
- **Таблиця дії переходу** для поточного сканування переходів, зафіксованих внизу кроків попередньої таблиці.

Стадія 3:

Дії, пов'язані з активними кроками, виконуються в такому порядку:

1. Дії на деактивацію кроків, які будуть деактивовані.
2. Дії на активацію кроків, які будуть ініційовані.
3. Неперервні дії активних кроків.

7.8.4 Кінцеві обчислення (post-processing)

Програма кінцевих обчислень вводиться у мовах Ladder Diagram, Instruction List чи Structured Text, і сканується повністю зверху донизу. Ці обчислення виконуються останніми перед активацією виходів і використовуються для програмування логічних виходів.

Дії, пов'язані з частиною Grafcet

Кінцеві обчислення використовуються для виконання дій, згенерованих при послідовних обчисленнях. Вони також використовуються для обробки виходів, ініційованих при послідовних обчисленнях.

Приклад (рисунок 7.24):

- **%I2.4**: захисне блокування для керування виходом %Q4.1.
- **%M26**: внутрішній біт керує дією і режимом зупинки.
- **%I1.0**: кнопковий перемикач.

Вихід %Q4.1 активізований кроками 5, 8 і 59 послідовних обчислень.

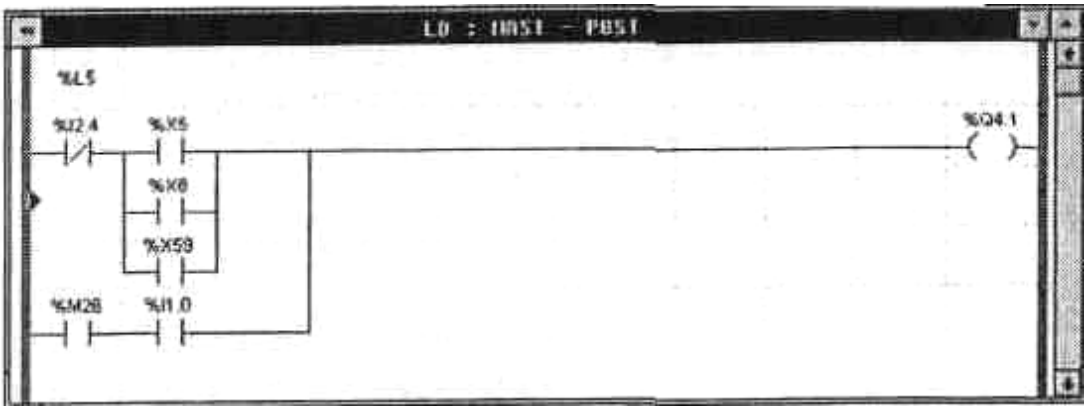


Рисунок 7.24 – Ілюстрація до прикладу

Дії, незалежні від частини Grafcet

Кінцеві обчислення також використовуються, щоб програмувати виходи, які є незалежними від послідовних обчислень.

Список питань для самоконтролю

1. Яка історія розвитку Grafcet?
2. Назвіть основні принципи Grafcet.
3. Які основні графічні символи в Grafcet?
4. Які об’єкти використовують в Grafcet?
5. Яка структура сторінки Grafcet?
6. Як здійснюється вибір послідовності в Grafcet?
7. Як використовуються з’єднувачі в Grafcet?
8. Які дії зв’язані з кроками в Grafcet?
9. Які умови зв’язані з переходами?
10. Як організовується ведуча задача в Grafcet?

Література

1. Schneider Automation Club. /Информационный бюллетень представительства Schneider Electric на Украине, №1, март 1997.
2. Schneider Automation Club. /Информационный бюллетень представительства Schneider Electric на Украине, №3, декабрь 1997.
3. Schneider Automation Club. /Информационный бюллетень представительства Schneider Electric на Украине, №5, июнь 1998.
4. Schneider Automation Club. /Информационный бюллетень представительства Schneider Electric на Украине, №6, январь 1999.
5. Schneider Automation Club. /Информационный бюллетень представительства Schneider Electric на Украине, №7, октябрь 1999.
6. PL7 Micro/Junior. Reference Manual.
7. PL7 Micro Software. Installation Manual for Application-specific Functions.

Додаток А
Системні біти

Біт	Функція	Початковий стан	Керування
%S0	1 – холодний старт (живлення повертається із втратою даних)	0	S чи U → S
%S1	1 – перезапуск із пам’яті (живлення повертається без втрати даних)	0	S чи U → S
%S4, %S5, %S6, %S7	Базовий час: 10 ms, 100 ms, 1 s, 1 min	-	S
%S8	Тестування під’єднаних датчиків (використовується на PLC, який не сконфігурований)	1	U
%S9	1 – виходи примусово переходять в аварійний режим для зміни конфігурації системи (fallback mode)	0	U
%S10	0 – помилка I/O	1	S
%S11	1 – переповнення сторожа	0	S
%S13	1 – перше сканування після установки в RUN	-	S
%S15	1 – помилка символьного рядка	0	S → U
%S16	0 – дефект в задачі I/O	1	S → U
%S17	Стан біта виведення протягом операції зсуву	0	S → U
%S18	1 – помилка арифметичного переповнення	0	S → U
%S19	1 – відхилення від установленого значення періоду задачі	0	S → U
%S20	1 – переповнення індексу	0	S → U
%S21	1 – ініціалізація Grafcet	0	S
%S22	1 – дезактивація Grafcet	0	S
%S23	1 – фіксування Grafcet	0	S
%S26	1 – переповнення ємності таблиці активних кроків в режимі STOP (див. %SW20 і %SW21)	0	S
%S30	1 – запуск головної задачі	1	U
%S31	1 – запуск швидкої задачі	1	U
%S38	1 – допуск подій	1	U
%S39	1 – насиченість в обробці подій	0	U

%S40 - %S47	0 – дефект у стійці з 0 до 7. Дефект у стійці – логічне АБО дефектів модуля стійки	1	S
%S49	1 – реактивація станів виведень	0	
%S50	1 – установка годинника в реальному масштабі часу	0	U
%S59	1 – дозволяє настроювання поточної дати	0	U
%S66	1 – дозволяє дисплей з 7 сегментами	0	U
%S67	0 – батарея карточки з пам’яттю у використанні	-	S
%S68	0 – резервна батарея (процесор) у використанні	-	S
%S69	1 – дозволяє режим візуального відображення пам’яті WORD на дисплеї	0	U
%S70	1 – обновлює слова обміну на TSX Nano	0	U $\rightarrow$ S
%S80	1 – скидує лічильник повідомлень	0	U $\rightarrow$ S
%S90	1 – обновлює загальні слова	0	S $\rightarrow$ U
%S100	Протокол термінального порту	-	S

S – керований системою
U – керований користувачем
U $\rightarrow$ S – установка в 1 користувачем, скидання в 0 системою
S $\rightarrow$ U – установка в 1 системою, скидання в 0 користувачем

Додаток Б
Системні слова

Слово	Функція	Керування
%SW0	Величина періоду головної задачі (періодичне виконання)	U
%SW1	Величина періоду швидкої задачі	U
%SW8	Перевірка читання входів для кожної задачі	U
%SW9	Перевірка поновлення виходів для кожної задачі	U
%SW10	Перше сканування після холодного старту %SW10:X0: головна задача, %SW10:X1: швидка задача, bit–0 – перше сканування, bit–1 кінець виконання	S
%SW11	Сторожовий час	S
%SW13	Основна адреса станції	S
%SW17	Тип дефекта виконання програми для дії з плаваючою точкою	
%SD18	Лічильник абсолютного часу	S і U
%SW20	Число активних кроків (64 максимум)	S
%SW21	Число переходов, що допускаються (96 максимум)	S
%SW30	Час виконання сканування останньої головної задачі	S
%SW31	Максимальний час сканування головної задачі	S
%SW32	Мінімальний час сканування головної задачі	S
%SW33	Час виконання сканування останньої швидкої задачі	S
%SW34	Максимальний час сканування швидкої задачі	S
%SW35	Мінімальний час сканування швидкої задачі	S
%SW48	Число оброблених подій	S
%SW49, %SW50, %SW51, %SW52, %SW53	Функція годинника реального часу: слова, що вміщують поточні значення дати і часу (в двійково-десятковому коді (BCD)), %SW49 – день тижня (тип дня), %SW50 – секунди, %SW51 – години і хвилини, %SW52 – місяць і день, %SW53 – століття і рік	S і U
%SW54, %SW55, %SW56, %SW57	Функція годинника реального часу: слова, що вміщують дату і час останнього збою живлення чи зупинку PLC (в двійково-десятковому коді (BCD)): %SW54 – секунди і код дефекта, %SW55 – години і хвилини, %SW56 – місяць і день, %SW57 – століття і	S

	рік	
%SW58	Ідентифікаційний код останньої зупинки і дня тижня (тип дня)	S
%SW59	Настроювання інкрементації поточної дати і дня	U
%SW66	Керування дисплеєм з 7 сегментами	U
%SW67 %SW68 %SW69	Керування режимом "Display": %SW67: читання кнопок, %SW68: поточні і максимальні індекси відображуваних об’єктів, %SW69: номер першого об’єкта у відображуваній зоні	S і U
%SW80	Кількість повідомлень, які послані системою до порту термінала	S і U
%SW81	Кількість повідомлень, що отримані системою від порту термінала	S і U
%SW82	Кількість повідомлень, які послані системою до PCMCIA-модуля	S і U
%SW83	Кількість повідомлень, які отримані системою від PCMCIA-модуля	S і U
%SW84	Кількість телеграм, які послані системою	S і U
%SW85	Кількість телеграм, що отримані системою	S і U
%SW86	Кількість повідомлень, від яких відмовляється система	S і U
%SW109	Лічильник кількості аналогових каналів, примусово встановлених в 0	S
%SW124	Тип останнього дефекта центрального процесора	S
%SW125	Тип дефекта блокування	S
%SW126	Адреса команди дефекта блокування	S

S – керований системою
U – керований користувачем

Навчальне видання

В.Ю. Кучерук, В.О. Поджаренко, П.І. Кулаков

Програмування логічних контролерів Schneider Electric

Навчальний посібник

Оригінал-макет підготовлено авторами

Редактор С.А.Малішевська

Підписано до друку
Формат 29,7х42 ¼ Гарнітура Times New Roman
Друк різографічний Ум. друк. арк.
Тираж __ прим.
Зам. №

Віддруковано в комп'ютерному інформаційно-видавничому центрі Вінницького державного технічного університету
21021, м. Вінниця, Хмельницьке шосе, 95, ВДТУ, ГНК, 9-й поверх
Тел. **(0432) 44-01-59**