МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

з виконання лабораторних робіт № 1, 2, 3

“ Автоматизація на основі контролерів S7-200 фірми SIEMENS “

з дисципліни «Програмно-технічні комплекси засобів автоматизації»

для студентів всіх форм навчання
освітньої програми «Промислова автоматика»
спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка

2024

Методичні вказівки з виконання лабораторних робіт № 1, 2, 3 "Автоматизація на основі контролерів S7-200 фірми SIEMENS" з дисципліни «Програмно-технічні комплекси засобів автоматизації» для студентів всіх форм навчання освітньої програми «Промислова автоматика» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка. / Укладачі: Кулинич Е.М., Осадчий В. В. – НУ «Запорізька політехніка», 2024. -86с.

Укладачі: Кулинич Е.М., к.т.н., доцент,
Осадчий В. В. к.т.н., доцент

Рецензент: А.В. Пирожок, к.т.н., доцент

Завідувач кафедрою ЕПА: А.В. Пирожок, к.т.н., доцент

Затверджено на засіданні кафедри ЕПА

Протокол № 1 від 10 серпня 2023 р.

Рекомендовано
на засіданні НМК ЕТФ
протокол № 1 від 21 вересня 2023 р.

ЗМІСТ

ПЕРЕДМОВА

Методичні вказівки містять опис трьох з шести лабораторних робіт з дисципліни «Програмно-технічні комплекси засобів автоматизації» у відповідності до навчальних планів ОКР бакалаврів освітньої програми «Промислова автоматика» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка, рекомендації до їх виконання.

Цикл лабораторних робіт №1-3 з дисципліни «Програмно-технічні комплекси засобів автоматизації» виконується на лабораторних стендах техніки малої автоматизації фірми Siemens, який детально описаний у лабораторній роботі №1 курса «Інтелектуальний аналіз даних»..

В лабораторних роботах наведені короткі теоретичні відомості, індивідуальні завдання та рекомендації щодо вводу у експлуатацію та налагодження систем керування, що дозволяє ознайомитися з можливостями, принципами програмування, режимами роботи, способами керування на основі на основі контролерів S7-200 фірми SIEMENS. Ознайомитися з програмним забезпеченням STEP 7 Micro/WIN фірми Siemens.

Усі лабораторні роботи розраховані на виконання протягом відведеного навчального часу за умови належної теоретичної підготовки.

У результаті попередньої підготовки потрібно вміти відповісти на всі контрольні запитання до роботи; під час лабораторних занять виконати необхідні завдання; оформити звіт про роботу; дати на підпис викладачеві. Звіти оформляти кожному студентові в окремому зошиті.

Для студентів всіх форм навчання освітньої програми «Промислова автоматика» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка».

1 ЛАБОРАТОРНА РОБОТА №1

АВТОМАТИЗАЦІЯ КОНВЕЄРА НА ОСНОВІ КОНТРОЛЕРА SIMATIC S7-200 фірми SIEMENS ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ STEP 7 MICRO/WIN

Мета: Оцінити обсяг робіт, які належить виконати в цьому курсі на цьому стенді. Зрозуміти палітру пристроїв автоматизації, ознайомитись з основними функціями контролера S7-200; Реалізувати програму керування транспортером.

Зрозуміти перспективи використання цього стенду в інших курсах, в дипломному проектуванні, а також у застосуванні отриманих на ньому навичок для вирішення завдань автоматизації і для модернізації існуючих систем управління.

1.1 Мікро ПЛК S7-200

Серія S7-200 – це ряд мікропрограмних логічних контролерів (мікроконтролерів), які можуть керувати різноманітними прикладними системами автоматизації. Компактна конструкція, низька вартість і потужна система команд роблять контролери S7-200 ідеальним засобом рішення для управління малими додатками. Велика різноманітність моделей S7-200 та інструментальні засоби програмування на основі Windows забезпечують необхідну гнучкість при вирішенні ваших завдань автоматизації.

Сімейство програмованих логічних мікроконтролерів (мікро-ПЛК) S7-200 може керувати широким спектром пристроїв для вирішення ваших завдань автоматизації.

S7-200 контролює входи і змінює виходи під керуванням програми користувача, яка може містити булеві логічні операції, функції рахунку і часу, складні математичні операції та операції з обміну даними з іншими інтелектуальними пристроями. Завдяки компактній конструкції, гнучкій конфігурації і потужного набору команд S7-200 у вищій мірі придатний для вирішення широкого спектру прикладних задач керування.

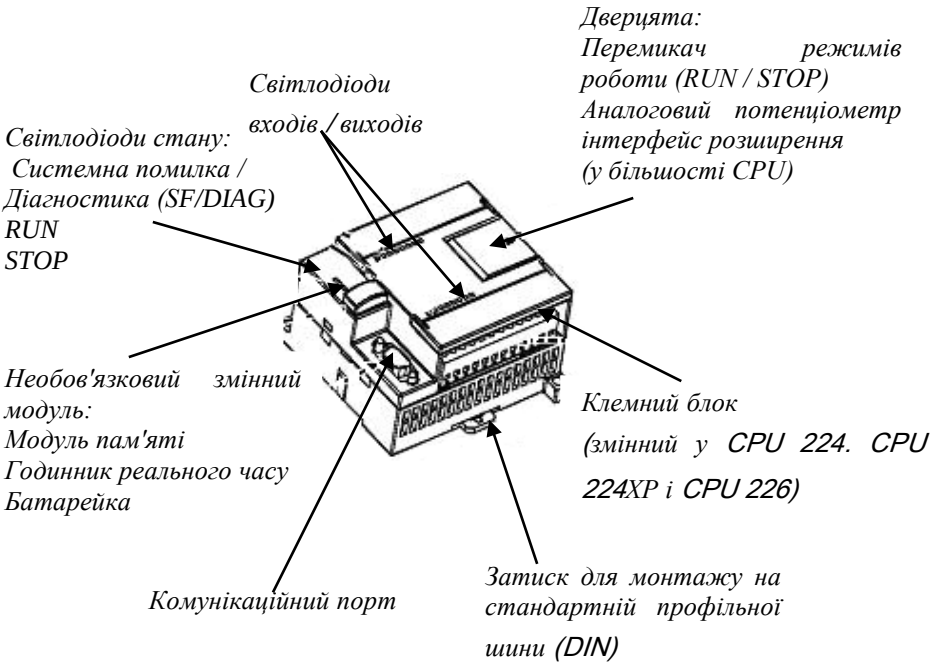


Рисунок 1.1 – Мікро ПЛК S7-200

1.2 Основи ПЛК

Серія S7-200 – це ряд мікропрограмних логічних контролерів (мікроконтролерів), які можуть керувати різноманітними прикладними системами автоматизації. Компактна конструкція, низька вартість і потужна система команд роблять контролери S7-200 ідеальним засобом рішення для управління малими додатками. Велика різноманітність моделей S7-200 та інструментальні засоби програмування на основі Windows забезпечують необхідну гнучкість при вирішенні ваших завдань автоматизації.

Сімейство програмованих логічних мікроконтролерів (мікро-ПЛК) S7-200 може керувати широким спектром пристроїв для вирішення ваших завдань автоматизації.

S7-200 контролює входи і змінює виходи під керуванням програми користувача, яка може містити булеві логічні операції, функції рахунку і часу, складні математичні операції та операції з обміну даними з іншими інтелектуальними пристроями. Завдяки компактній конструкції, гнучкій конфігурації і потужного набору команд S7-200 у вищій мірі придатний для вирішення широкого спектру прикладних задач керування.

Основною функцією S7-200 є контроль польових входів і на основі логіки управління, включення і виключення польових вихідних пристроїв. У цьому розділі пояснюються основи виконання програми, різні види використовуваної пам'яті і способи збереження.

1.2.1 Виконання логіки управління за допомогою S7-200

S7-200 обробляє логіку управління у вашій програмі циклічно, зчитуючи і записуючи дані.

S7-200 ставить вашу програму у відповідність фізичним входів і виходів

Основний принцип дії S 7-200 дуже простий.

S7-200 зчитує стан входів

Програма, що зберігається в S7-200, використовує ці входи для аналізу логіки керування. Під час обробки програми S7-200 оновлює дані.

S7-200 записує дані на виходи.

На рис. 2.2 показано зв'язок між простою комутаційною схемою і S7-200. У цьому прикладі стан вимикача для запуску двигуна логічно пов'язаний зі станами інших входів. Оцінки цих станів визначають потім сигнальний стан виходу для виконавчого пристрою, що запускає двигун.

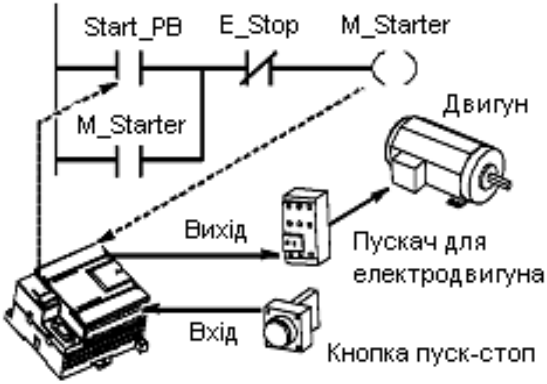


Рисунок 1.2 – Керування входами та виходами

1.2.2 Виконання завдань в циклі

S7-200 виконує послідовність завдань неодноразово. Ця регулярна обробка завдань називається циклом. Як показано на рис. А-2, S7-200 виконує в циклі більшість або всі з наступних завдань:

- читання входів: S7-200 копіює стан фізичних входів в регістр входів образу процесу;

- виконання логіки управління у програмі: S7-200 виконує команди програми і зберігає значення в різних областях пам'яті;
- обробка запитів на обмін даними: S7-200 виконує всі завдання, необхідні для обміну даними;
- самодіагностика CPU: S7-200 перевіряє, щоб вбудоване програмне забезпечення, програмна пам'ять і всі модулі розширення працювали належним чином;
- запис у виходи: Значення, які у регістрі виходів образу процесу, записуються у фізичні виходи.

Виконання програми користувача залежить від того, чи знаходиться S7-200 в стані STOP або в стані RUN. У стані RUN ваша програма виконується; в стані STOP ваша програма не виконується.

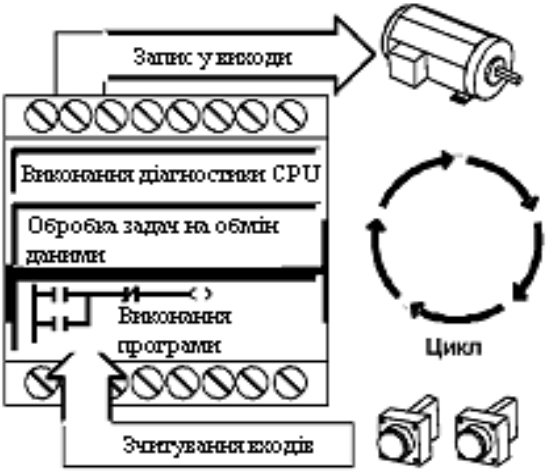


Рисунок 1.3 – Цикл S7-200

1.2.3 Читання входів

Цифрові входи: На початку циклу поточні значення цифрових входів зчитуються, а потім записуються в регістр входів образу процесу.

Аналогові входи: S7-200 не оновлює аналогові входи модулів розширення автоматично як частина циклу, якщо ви не активізували фільтрацію аналогових входів. Аналоговий фільтр забезпечує стабільність сигналів. Ви можете активізувати аналоговий фільтр для кожного входу.

Якщо фільтр для аналогового входу активізований, то S7-200 оновлює цей аналоговий вхід один раз за цикл, виконує функцію фільтрації та зберігає відфільтроване значення всередині. Це відфільтроване значення потім надається в розпорядження кожного разу, коли ваша програма звертається до цього аналоговому входу.

Якщо фільтр аналогового входу вимкнений, то S7-200 зчитує значення цього аналогового входу з модуля розширення щоразу, коли ваша програма звертається до аналогового входу.

Аналогові входи AIW0 і AIW2 модуля CPU 224XP оновлюються в кожному циклі самими останніми результатами аналого-цифрового перетворювача. Цей перетворювач працює з середніми значеннями (sigma-delta), і ці значення зазвичай не потребують програмної фільтрації.

Порада: Фільтр аналогового входу забезпечує стабільність аналогових значень. Фільтр аналогового входу слід активізувати в додатках, в яких вхідний сигнал повільно змінюється з часом. Якщо мова йде про швидко мінливому сигналі, то аналоговий фільтр активізувати не слід.

Не застосовуйте аналоговий фільтр у модулях, які передають цифрові дані або сигнали тривоги в аналогових словах. Завжди вимикайте аналоговий фільтр для провідних модулів з RTD, термопарами і AS-інтерфейсом.

1.2.4 Виконання програми

На цьому етапі циклу S7-200 обробляє програму з першої команди до останньої. Ви можете безпосередньо керувати входами і виходами і отримувати, таким чином, доступ до них під час виконання основної програми чи програми обробки переривань.

Якщо ви використовуєте в своїй програмі переривання, то програми обробки переривань, які ставляться у відповідність перериваються подіям, зберігаються як час основної програми. Однак програми обробки переривань виконуються не як складова частина нормального циклу, а тільки тоді, коли відбувається переривання події (воно можливе в будь-якому місці циклу).

1.2.5 Обробка запитів на обмін даними

На ділянці циклу, виділеному для обробки комунікацій, S7-200 обробляє всі повідомлення, отримані з комунікаційного порту або від інтелектуальних модулів вводу/виводу.

1.2.6 Самодіагностика CPU

На цій ділянці циклу S7-200 перевіряє належну роботу CPU області пам'яті і стан модулів розширення.

1.2.7 Запис на цифрові виходи

У кінці кожного циклу S7-200 записує значення, які зберігаються в регістрі виходів образу процесу, в цифрові виходи. (Аналогові виходи оновлюються негайно, незалежно від циклу).

1.3 Доступ до даних S7-200

S7-200 зберігає інформацію в різних місцях пам'яті, доступу до даних S7-200.

S7-200 зберігає інформацію в різних місцях пам'яті, які мають однозначні адреси. Ви можете явно вказати адресу в

пам'яті, до якого ви хочете звернутися. Завдяки цьому ваша програма має прямий доступ до інформації. Таблиця 1.1 показує діапазон цілих значень, які можуть бути представлені за допомогою даних різної довжини.

Таблиця 1.1. – Десяткові і шістнадцятирічні діапазони для даних різної довжини

Подання	**Байт (B)**	**Слово (PM)**	**Подвійне слово (P)**
Ціле без знаку	від 0 до 255 від 0 до FF	від 0 до 65 535 від 0 до FFFF	від 0 до 4294967295 від 0 до FFFFFFFF
Ціле зі знаком	від -128 до + 127 від 80 до 7F	від -32768 до +32767; від 8000 до 7FFF	від -2147483648 до 2147483647 від 8000 0000 до 7FFF FFFF
Речовинне IEEE 32-бітове з плаваючою точкою	Не підходить	Не підходить	від +1.175495 E-38 до +3.402823 E +38 (Позитивне) від-1.175495E-38 до-3.40282EE +38 (негативне)

Для звернення до біту в деякій області пам'яті ви повинні вказати адресу біта. Ця адреса складається з ідентифікатора області пам'яті, адреси байта і номери біта. На рис. 1.4 показаний

приклад звернення до біту (адресація в форматі «байт.біт»), У цьому прикладі за областю пам'яті та адресою байта (I = input [вхід], 3 = байт 3) слід точка {«.»). щоб відокремити адресу біта (біт 4).

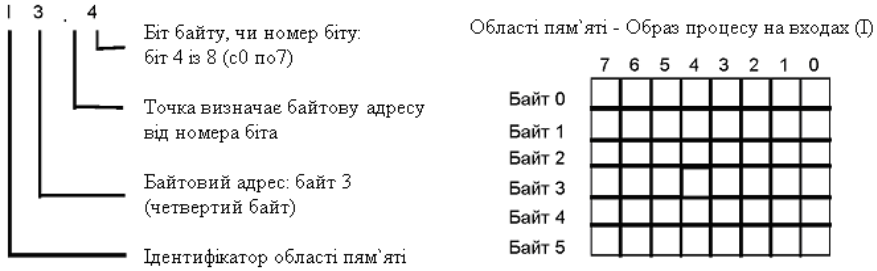


Рисунок 1.4 – Адресація байт.біт

Застосовуючи формат байт.біт, ви можете звернутися до даних у більшості областей пам'яті (V, I, 6. M, E, I. і EM) як до байтам, словам або подвійним словами. Якщо ви хочете звернутися до байту, слову або подвійному слову даних у пам'яті, то ви повинні вказати ці адреси подібно адресою біта. Ви вказуєте ідентифікатор області, позначення довжини даних і початкову адресу байта, слова чи подвійного слова, як показано на рис.1.5.

До даних в інших областях пам’яті (на прикл. T, TC, HC і акумулятори), ви звертаєтесь вказуючи в якості адреси ідентифікатор області і номер елементу.

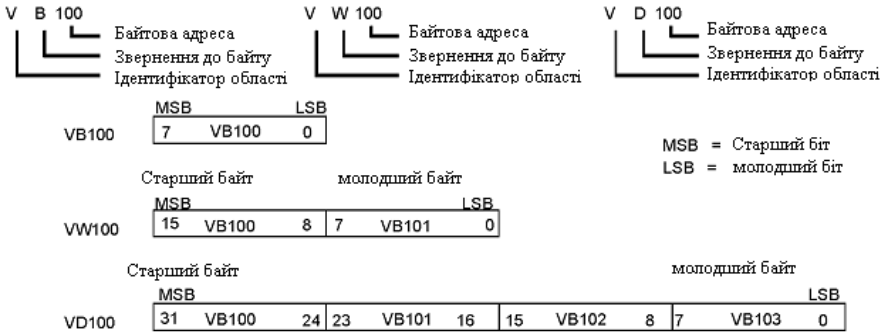


Рисунок 1.5 – Звернення до однієї і тієї ж адреси в форматі байту, слова і подвійного слова

1.4 Область пам'яті даних

1.4.1 Регістр входів образу процесу: I

На початку кожного циклу S7-200 опитує фізичні входи і записує отримані значення в регістр входів образу процесу. До образу процесу можна звернутися у форматі біта, байта, слова та подвійного слова:

Біт: *[адрес байта].[адрес бита]* I0.1

Байт, слово або подвійне слово: *[довжина] [початковий адресу байта]* IB4

1.4.2 Регістр виходів образу процесу: Q

В кінці циклу S7-200 копіює значення, які зберігаються в регістрі виходів образу процесу, у фізичні виходи. До образу процесу можна звернутися у форматі біта, байта, слова та подвійного слова:

Біт: Q*[адреса байта].[адреса біта]* Q1.1

Байт, слово або подвійне слово: Q*[довжина] [початкова адреса байта]* QB5

1.4.3 Область пам'яті змінних: V

Пам'ять змінних можна використовувати для зберігання проміжних результатів операцій, виконуваних у вашій програмі. У пам'яті змінних ви можете зберігати також інші дані, що мають відношення до процесу або до вирішення вашого завдання автоматизації. До пам'яті змінних можна звернутися у форматі біта, байта, слова та подвійного слова:

Біт: V*[адреса байта].[адреса біта]* V10.2

Байт, слово або подвійне слово: V*[довжина].[початкова адреса байта]* VW100

1.4.4 Область бітової пам'яті: M

Біти пам'яті (меркери) можна використовувати як керуючі реле для зберігання проміжних результатів операцій або іншої керуючої інформації. До бітів пам'яті можна звернутися у форматі біта, байта, слова та подвійного слова:

Біт: M*[адреса байта].[адреса біта]* M26.7

Байт, слово або подвійне слово: M*[довжина].[початкова адреса байта]* MD20.

1.5 Лічильники: C

S7-200 має в своєму розпорядженні три види лічильників, які підраховують наростаючі фронти на рахункових входах

лічильника; один вид лічильників веде прямий рахунок, інший вважає тільки у зворотному напрямку, а третій вид вважає в обох напрямках. З лічильником пов'язані дві змінні:

Поточне значення: це 16-бітове ціле зі знаком зберігає рахункове значення, накопичене лічильником.

Біт лічильника: цей біт встановлюється або скидається, коли поточне значення стає рівним передвстановленому значенню. Передустановлене значення вводиться як частина команди лічильника.

Ви звертаєтеся до обох цих елементів даних через адресу лічильника (C + номер лічильника). Чи відбувається звернення до біту лічильника або до поточного значення, залежить від використовуваної команди: команди з операндами в бітовому форматі звертаються до біту лічильника, тоді як команди з операндами в форматі слова звертаються до поточного значення. Як показано на рис.1.6, команда "Нормально відкритий контакт" звертається до біту лічильника, а команда "Передати слово" звертається до поточного значення лічильника.

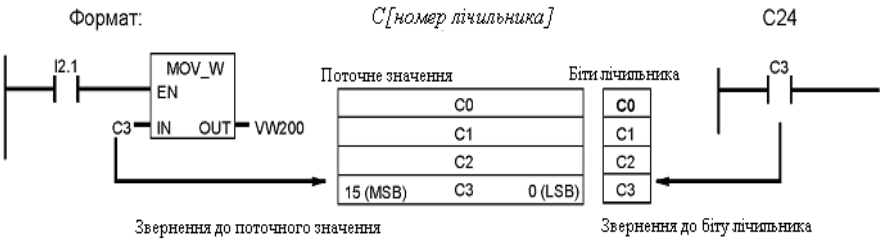


Рисунок 1.6 – Звернення до біту або до поточного значення лічильника

1.5.1Швидкісні лічильники: HC

Швидкісні лічильники підраховують швидкі події незалежно від циклу CPU. Швидкісні лічильники мають у своєму

розпорядженні 32-бітове ціле рахункове значення (поточне значення). Для звернення до рахункового значення швидкісного лічильника введіть його адресу, вказавши область пам'яті (НС) і номер лічильника (напр., НСО). Поточне значення швидкісного лічильника захищено від запису і може бути адресоване тільки у форматі подвійного слова (32 біта).

Формат: НС *[номер швидкісного лічильника]* НС1

1.6 Акумулятори: АС

Акумулятори - це елементи читання/запису, які можуть використовуватися як пам'ять. Наприклад, ви можете використовувати акумулятори для передачі параметрів у підпрограми і з них або для зберігання проміжних результатів розрахунків. S7-200 має в своєму розпорядженні чотири 32-бітових акумулятора (АФО, АС1, АС2 і АСЗ). До даних в акумуляторах можна звернутися у форматі біта, слова чи подвійного слова.

Довжина даних, до яких проводиться звернення, залежить від команди, яка використовується для звернення до акумулятора. Як показано на рис.1.7, при зверненні до акумулятора у форматі біта або слова використовуються молодші 8 або 16 бітів значення, що зберігається в акумуляторі. При зверненні до акумулятора у форматі подвійного слова використовуються всі 32 біта.

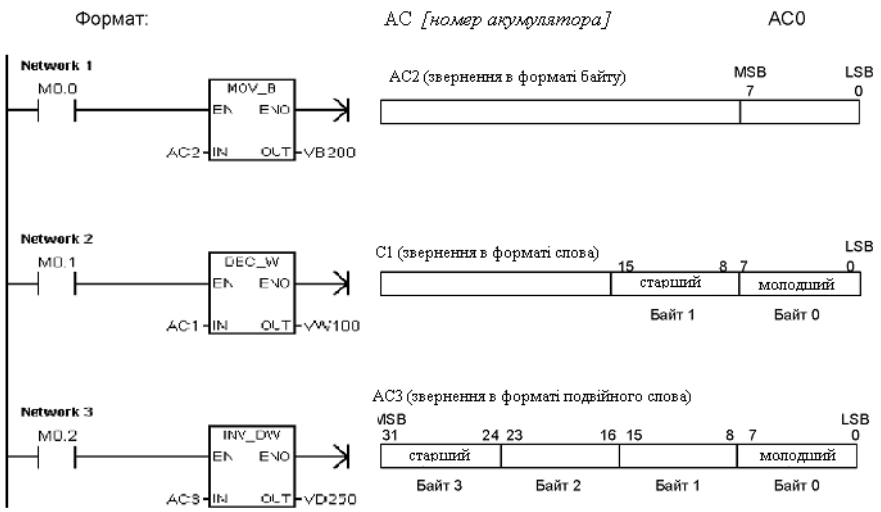


Рисунок 1.7 – Звернення до акумуляторів

1.7 Спеціальні біти пам'яті: SM

Спеціальні біти пам'яті (SM) надають засіб для обміну даними між CPU і вашою програмою. Ви можете використовувати ці біти для вибору та управління деякими спеціальними функціями CPU S7-200, наприклад: біт, який встановлюється тільки в першому циклі; біт, який встановлюється і скидається з фіксованою частотою, або біт, який вказує на стан арифметичної чи іншої команди. До SM-бітам можна звертатися у форматі біта, слова чи подвійного слова:

Біт: SM *[адреса байта]. [адреса біта]* SM0.1

Байт, слово або подвійне слово: SM *[довжина].[початкова адреса байту]* SMB86

1.8 Пам'ять локальних даних: L

S7-200 має в своєму розпорядженні 64 байти локальної пам'яті, з яких 60 можуть бути використані в якості проміжної пам'яті або для передачі формальних параметрів у підпрограми.

Порада! При програмуванні в LAD або FBD останні чотири байти зарезервовані для STEP 7-Micro/WIN.

Пам'ять локальних даних схожа на пам'ять змінних з однією істотною відмінністю. Пам'ять змінних доступна глобально, тоді як пам'ять локальних даних доступна локально. Глобальна доступність означає, що до адреси в цій області пам'яті можна звернутися з будь-якої організаційної одиниці програми (з основної програми, підпрограми або підпрограм обробки переривань). Локальна доступність означає, що ця область пам'яті ставиться у відповідність певної організаційної одиниці програми. S7-200 виділяє 64 байти локальної пам'яті для головної програми, 64 байти для кожного рівня вкладеності підпрограм і 64 байта для програм обробки переривань.

До області локальних даних, поставленої у відповідність основній програмі, не мають доступу підпрограм і програми обробки переривань. Підпрограма не може звертатися до області локальних даних основної програми, програми обробки переривань або іншої підпрограми. Аналогічно, програма обробки переривань не має доступу до області локальних даних основної програми або підпрограми.

S7-200 виділяє область локальних даних у міру необхідності. Це означає, що при виконанні основної програми області локальних даних для підпрограм і програм обробки переривань не існують. Якщо виникає переривання або викликається підпрограма, то за потреби виділяється локальна пам'ять. Знову виділена локальна пам'ять може знову використати ті самі адреси, які використовувалися іншою підпрограмою або програмою обробки переривань.

S7-200 не ініціалізує область локальних даних до моменту її призначення, тому вона може містити будь-які значення. Якщо

при виклику підпрограми передаються формальні параметри, то S7-200 зберігає значення переданих параметрів у відповідних адресах області локальних даних, виділених цією підпрограмою. Адреси в області локальних даних, які не отримали значень при передачі формальних параметрів, не ініціалізуються і при виділенні можуть містити довільні значення.

Біт: *[адреса байтів], [адресу біта]* L0.0

Байт, слово або подвійне слово: *[довжина] [початкова адреса байта]* LB33

1.9 Аналогові виходи: AQ

S7-200 перетворює цифрові величини, що мають довжину слова (16 бітів), у струм або напруга пропорційно цифровий величиною. Звернення до цих значень проводиться через ідентифікатор області (AT), довжину даних (W) і початкову адресу байта. Так як у випадку аналогових виходів мова йде про слова, які завжди починаються на байтах з парними номерами (наприклад, 0. 2, 4 і т.д.), то ці значення записуються з адресами парних байтів (наприклад, AQWO, AQW2, AQW4). Аналогові виходи можна тільки записувати.

Формат: AQW *[начальний адресу байта]* AQW4

1.10 Реле керування черговістю (SCR): S

SCR або S-біти поділяють функціонування установки на окремі кроки чи еквівалентні частини програми. За допомогою реле керування черговістю програма керування представляється у вигляді структури, що складається з логічних сегментів. До S-бітам можна звертатися у форматі біта, слова чи подвійного слова.

Біт: S *[адреса байта].[адреса біта]* S3.1

Байт, слово або подвійне слово: S *[довжина].[початкова адреса байта]* SB4

1.11Формат дійсних чисел

Дійсні числа (або числа з плаваючою точкою) представляються як 32-бітові числа одноразової точності, формат яких описаний в стандарті ANSI / IEEE 754-1985. Див. рис.1.8. Звернення до речових чисел проводиться у форматі подвійного слова.

У S7-200 числа з плаваючою точкою мають точність до 6 десяткових розрядів. Тому при введенні константи з плаваючою точкою можна вказувати до 6 десяткових розрядів.

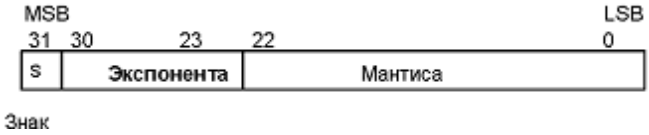


Рисунок 1.8 – Формат дійсного числа

1.11.1 Точність при обчисленнях з речовими числами

Розрахунки, що включають в себе довгі послідовності значень, що містять дуже великі і дуже малі числа, можуть призвести до неточних результатів. Це може статися, якщо числа відрізняються один від одного в 10 в ступені x разів, де x> 6. Наприклад:

100 000 000 + 1 = 100 000 000

1.12 Формат для рядків

Рядок - це послідовність символів, причому кожен символ зберігається як байт. Перший байт рядка визначає її довжину, тобто у кількості наявних у ній символів. На рис. 2.9 показаний формат рядка. Рядок може включати в себе від 0 до 254 символів, плюс байт, що містить інформацію про довжину, таким чином, максимальна довжина рядка дорівнює 255 байтам. Строкова константа обмежена 126 байтами

Рисунок 1.9 – Формат для рядків

1.13 Завдання постійного значення для команд S7-200

У багатьох командах для S7-200 можна використовувати константи. Константи можуть бути байтами, словами або подвійними словами. S7-200 зберігає всі константи у вигляді двійкових чисел, які можуть бути представлені в десятковому, шістнадцятковому форматі, у форматі ASCII або у форматі дійсних чисел (чисел з плаваючою точкою). Див. таблицю 1.2.
Таблиця 1.2 – Представлення постійних величин

Представлення	Формат	Приклад
Десяткове	[десяткове значення]	20047
Шістнадцяткове	16 # [шістнадцяткове значення]	16 # 4E4F
Двійкове	2 # [двійкове число]	2# 1010 0101 1010 0101
ASCII	'[текст ASCII]'	'ABCD'
Речове	ANSI / IEEE 754-1985	+1.175495 E-38 (позитивне) -1.175495E-38 (негативне)
Рядок	«[текст рядка]»	«ABCDE»

Порада! У CPU S7-200 не можна вказувати конкретні типи даних (коли ви, наприклад, хочете вказати, що константа повинна бути збережена як ціле число (16 бітів), ціле число зі знаком чи подвійне ціле (32 біта). Наприклад, команда додавання може використовувати значення, що зберігається в VW100, як ціле число зі знаком, а команда "Що виключає АБО" те ж саме значення в VW100 може використовувати як двійкове значення без знака.

1.14 Робота в середовищі STEP 7-Micro/WIN

Клацніть на символі STEP 7-Micro/WIN, щоб відкрити новий проект. На рис. 1.10 показано новий проект.

Зверніть увагу на навігаційну панель. За допомогою символів на навігаційній панелі ви можете відкривати окремі елементи проекту STEP 7-Micro/WIN.

Клацніть на символі Communications на навігаційній панелі, щоб викликати діалогове вікно "Communications [Обмін даними)". Це діалогове вікно використовується для установки зв'язків для STEP 7-Micro/WIN.

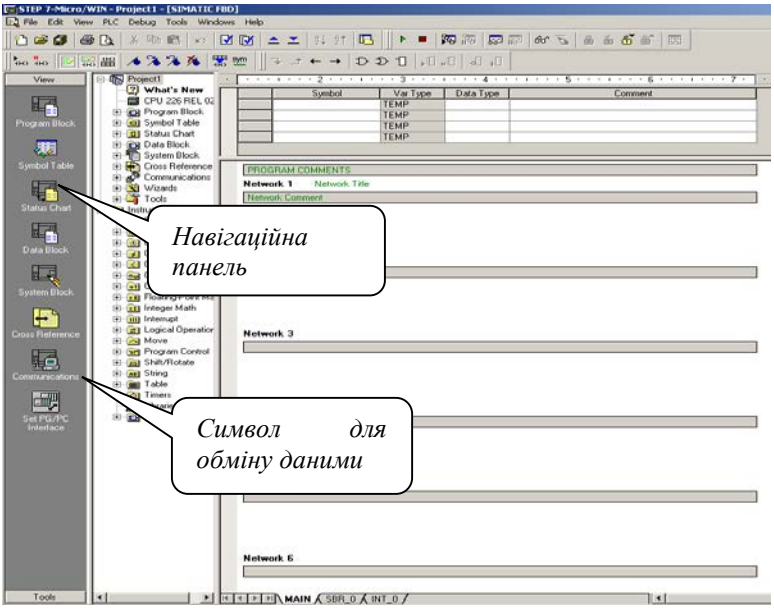


Рисунок 1.10 – Новий проект Step 7-Micro/WIN

1.15 Редактор програм

Щоб відкрити редактор програм, клацніть на символі Program Block [Програмний блок]. Див. рис. 1.11.

Зверніть увагу на дерево команд і редактор програм. Дерево команд використовується для вставки команд контактного плану (LAD) у сегменти редактора програм шляхом буксирування команд за допомогою миші з дерева команд у сегменти.

Символи на панелі інструментів надають можливість швидкого виклику команд меню.

Після введення і збереження програми ви можете завантажити її в S7-200.

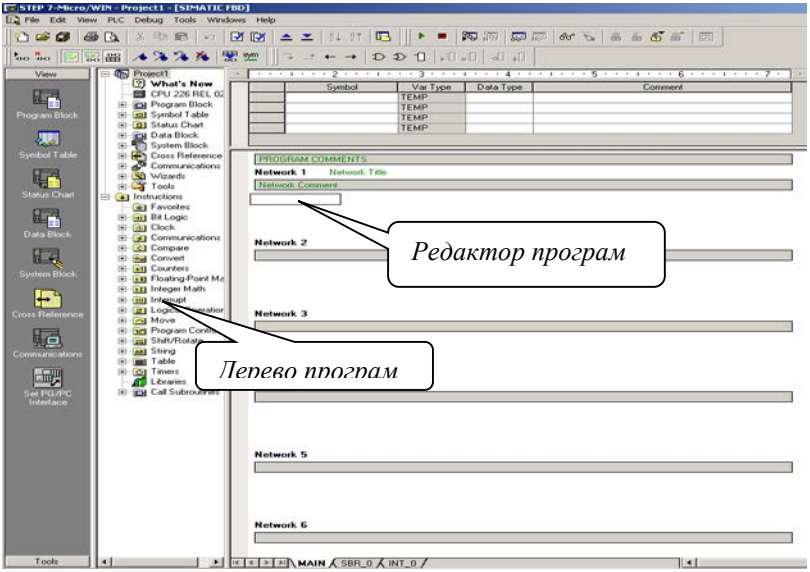


Рисунок 1.11 – Вікно Step 7-Micro/WIN

1.16 Команди STEP 7 MicroWIN

1.16.1 Бітові логічні операції

Контакти. Стандартні контакти

Команди "Нормально відкритий контакт" (LD. A і O) і "Нормально замкнутий контакт" (LDN. AN, ON) отримують початкове значення з пам'яті або з регістра образу процесу. Стандартні контакти отримують початкове значення з пам'яті (або з регістра образу процесу, якщо типом даних є I або Q).

Нормально відкритий контакт замкнутий (включений), коли біт дорівнює 1, а нормально замкнутий контакт замкнутий (включений), коли біт дорівнює 0. У FBD до блоків І та АБО може бути підключено не більше 32 входів. У STL команди, що представляють нормально відкритий контакт, завантажують значення адресного біта в вершину стека, або виконують логічне сполучення значення адресного біта зі значенням у вершині стека згідно з таблицею істинності логічного І чи АБО, а команди, що представляють нормально замкнутий контакт, завантажують логічне заперечення значення адресного біта в вершину стека або виконують логічне поєднання логічного заперечення значення адресного біта зі значенням у вершині стека згідно з таблицею істинності логічного І чи АБО.

Контакти. Безпосередньо керовані контакти

Безпосередньо керований контакт при своїй актуалізації не залежить від циклу S7-200, його значення оновлюється негайно. Команди "Безпосередньо керований нормально відкритий контакт" (LDI, AI і OI) і "Безпосередньо керований нормально замкнутий контакт" (LDNI, ANI і ONI) при виконанні команди отримують значення фізичного входу, однак, регістр образу процесу не оновлюється.

Безпосередньо керований нормально відкритий контакт замкнутий (включений), коли фізичний вхід (біт) знаходиться в стані 1, а безпосередньо керований нормально замкнутий контакт замкнутий (включений), коли фізичний вхід (біт) знаходиться в стані 0. Команди, що представляють безпосередньо керований нормально відкритий контакт, безпосередньо завантажують значення фізичного входу у вершину стека або виконують логічне сполучення значення фізичного входу зі значенням у вершині стека згідно з таблицею істинності логічного І чи АБО, а команди, що представляють безпосередньо нормально керований замкнутий контакт, безпосередньо завантажують логічне заперечення значення фізичного входу у вершину стека або виконують логічне сполучення заперечення значення фізичного входу зі значенням у вершині стека згідно з таблицею істинності логічного І чи АБО.

Команда NOT [HI]

Команда заперечення (NOT) змінює стан входу потоку сигналу (тобто вона змінює значення у вершині стека з 0 на 1 або з 1 на 0).

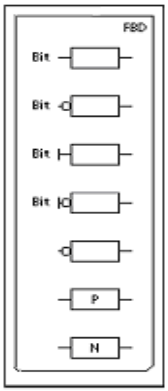


Рисунок 1.12 – Зображення блоків в FBD

1.16.2 Позитивний і негативний фронт

Контакт "Позитивний фронт" (EU) пропускає потік сигналу протягом одного циклу при кожній появі позитивного фронту. Контакт "Негативний фронт" (ED) пропускає потік сигналу протягом одного циклу при кожній появі негативного фронту. У команди "Позитивний фронт" при виявленні переходу значення у вершині стека з 0 на 1 значення у вершині стека встановлюється в 1; в іншому випадку воно встановлюється в 0. У команди "Негативний фронт" при виявленні переходу значення у вершині стека з 1 на 0 значення у вершині стека встановлюється в 1, в іншому випадку воно встановлюється в 0.

1.16.3 Котушки. Команди присвоєння

Команда присвоювання (=) записує нове значення для вихідного біта в регістр образу процесу. При виконанні команди присвоювання S7-200 встановлює або скидає вихідний біт в регістрі образу процесу. У LAD і FBD вказаний біт встановлюється рівним потоку сигналу. BSTL значення, що знаходиться у вершині стека, копіюється у зазначений біт.

1.16.4 Команди безпосереднього присвоювання бітового значення

Команда безпосереднього привласнення бітового значення (= 1) при своєму виконанні записує нове значення як у фізичний вихід, так і в образ процесу.

Коли виконується команда безпосереднього привласнення бітового значення, фізичний вихід (біт) негайно встановлюється відповідно до стану потоку сигналу. У STL команда безпосереднього привласнення бітового значення безпосередньо

копіює значення, що знаходиться у вершині стека, у зазначений фізичний вихід. Символ "I" означає безпосередній доступ; при виконанні команди нове значення записується у фізичний вихід і у відповідну клітинку регістру образу процесу. Тут є відмінність від інших видів доступу, які записують нове значення тільки в регістр образу процесу.

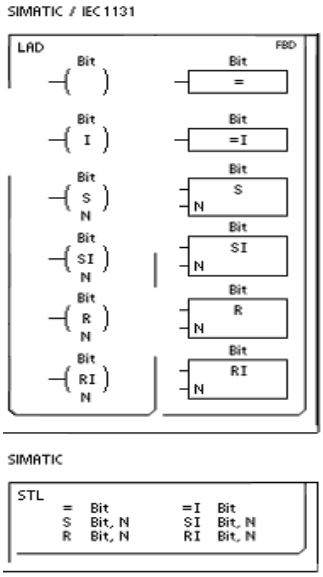


Рисунок 1.13 – Зображення котушок в FBD і STL

1.16.5 Команди установки і скидання

Команди установки (S) і скидання (R) встановлюють (включають) або скидають (вимикають) вказану кількість входів або виходів (N), починаючи із зазначеної адреси (біта). Ви можете встановити або скинути від 1 до 255 входів і виходів.

Якщо команда скидання вказує на біт таймера (T) або лічильника (C), то команда скидає біт таймера або лічильника і стирає поточне значення таймера або лічильника.

Збійні стани, що встановлюють ENO = O

- 0006 (непряма адреса)
- 0091 (операнд вийшов за межі допустимого діапазону).

1.16.6 Команди безпосередньої установки і безпосереднього скидання

Команди безпосередньої установки і безпосереднього скидання безпосередньо встановлюють (включають) або безпосередньо скидають (вимикають) вказану кількість входів або виходів (N), починаючи із зазначеної адреси (біта). Ви можете безпосередньо і негайно встановити або скинути від 1 до 128 входів і виходів. Символ "I" означає безпосередній доступ; при виконанні команди нове значення записується у фізичний вихід і у відповідну клітинку регістру образу процесу. Тут є відмінність від інших видів доступу, які записують нове значення тільки в регістр образу процесу.

1.16.7 Команди які реалізуються в функціональному блоці з двома стійкими станами: перевага установки і перевага скидання

Функціональний блок з двома стійкими станами і перевагою установки є тригер, у якого домінує установка. Якщо сигнал установки (S1) і сигнал скидання (R) одночасно приймають значення істина, то вихід (OUT) приймає значення істина.

Функціональний блок з двома стійкими станами і перевагою скидання представляє собою тригер, у якого домінує скидання. Якщо сигнал установки (S) і сигнал скидання (R1) одночасно приймають значення істина, то вихід (OUT) приймає значення брехня.

Параметр Bit представляє собою логічний параметр, який встановлюється або скидається. Додатковий вихід відображає сигнальний стан параметра Bit.

У таблиці 1.3 представлені стани функціональних блоків для програми-прикладу.

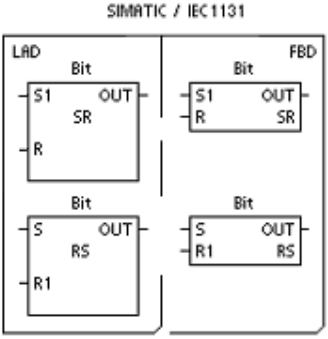
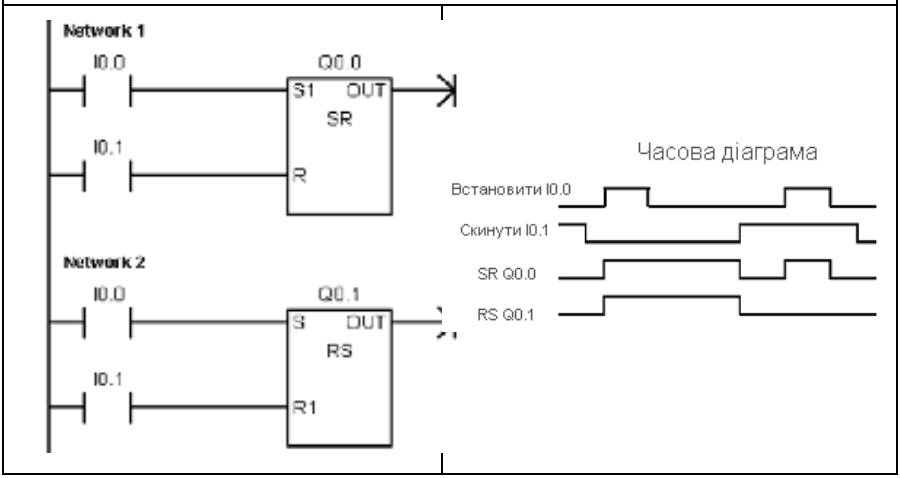


Рисунок 1.14 – Зображення функціонального блока с двома стійкими станами в FBD і LAD

Таблиця 1.3 – Допустимі операнди для функціонального блоку з двома стійкими станами

Входи/виходи	Тип даних	Операнди
S1, R	BOOL	I, Q, V, M, SM, S, T, C, потік сигналу
S, R1, OUT	BOOL	I, Q, V, M, SM, S, T, C, L, потік сигналу
Bit	BOOL	I, Q, V, M, S

Приклад: Функціональний блок з двома стійкими станами: перевага установки і перевага скидання



Таблиця 1.4 – Таблиця станів для функціональних блоків з двома стійкими станами перевага установки і перевага скидання

Команда	**S1**	**R**	**Out (Bit)**
Функціональний блок з двома стійкими станами: перевага установки (SR)	0	0	Попередній стан
	0	1	0
	1	0	1
	1	1	1
Команда	**S**	**R1**	**Out (Bit)**
Функціональний блок з двома стійкими станами: перевага установки (RS)	0	0	Попередній стан
	0	1	0
	1	0	1
	1	1	0

1.17 Хід виконання лабораторної роботи

Для виконання даної лабораторної роботи необхідно виконати наступні дії:

1. Зробити перекомутацію комутатора, налаштувавши на дозування (за допомогою програми AccessPort: D \ DA_stend \ connect \ _AccessPort).

2. Відкрити модель дозування (на комп'ютері для моделювання: D \ DA_stend \ models \ 2_LiquidDozing \ LiquidDozing.mdl).

3. Відкрити програму STEP 7 MicroWIN, для програмування конвеєра (на комп'ютері для програмування).

1.17.1 Налаштування стенду і комп'ютерів на роботу

Всі зміни комутацій на стенді не вимагають зміни існуючої схеми з'єднань стенду. Всі зміни настроюються програмно, шляхом коригування шаблонної таблиці комутацій (таблиця Excel) відповідно до вимоги лабораторних робіт. У принципі є можливість з'єднати будь-який цифровий вихід будь-якого пристрою стенду (кнопки, перемикачі, цифрові виходу контролера і STEP 7 MicroWIN, цифрові виходу приводів) з будь-яким цифровим входом пристроїв стенду (лампочки, цифрові входу контролера і STEP 7 MicroWIN, цифрові входу приводів).

Запускаємо програму AccessPort (рис. 1.15). Потім відкриваємо порт, в панелі інструментів: Tools / Port Switch. Для передачі вмісту результату таблиці комутації в комутатор в панелі інструментів зайти: Tools / Transfer file. У вікні натиснути кнопку Select file і вибираємо файл комутації (D: \ DA_stend \ connect \ 2_2_Doz_S7-200 \ 20070913_Коммутація_Дозіровка з фарбами_S7-200.txt.). Після натиснути кнопку "Send", при цьому

вміст буфера передається по послідовному інтерфейсу в мікропроцесорний комутатор стенду і виконує його перекомутацію. Нова комутація, буде автоматично виконана за переданими даними мікропроцесорним комутатором стенду. Якщо посилка пройшла успішно, то комутатор відповідає знаком точки, яка з'являється в полі буфера прийому.

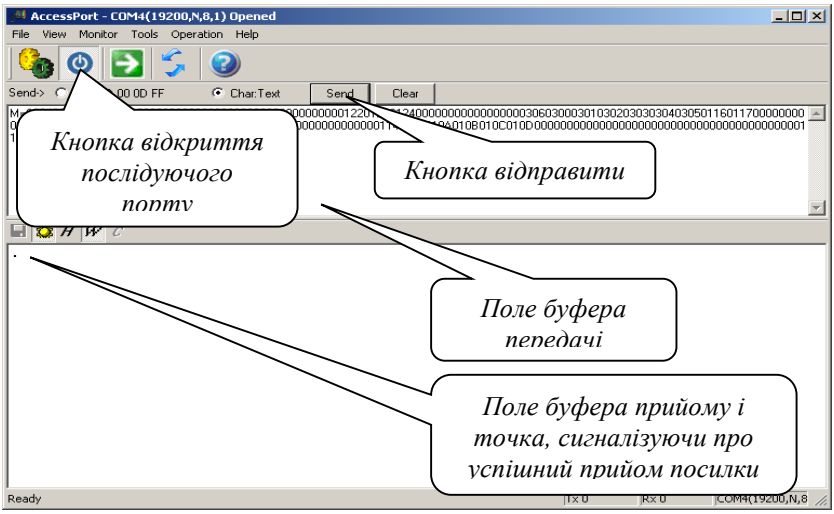


Рисунок 1.15 – Вікно програми AccessPort

На рис. 1.16 показані необхідні налаштування порту через який відбувається зв'язок з комутатором стенду.

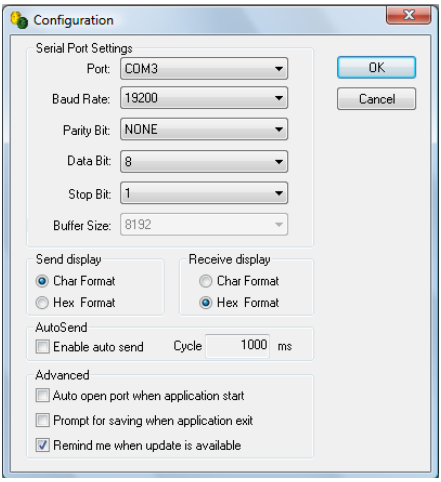


Рисунок 1.16 – Налаштування порта AccessPort

Після налаштування AccessPort необхідно запустити модель (Simulation / Start). Адреса моделі: (D: \ DA_stend \ models \ 2_LiquidDozing \ LiquidDozing.mdl).

1.17.2 Програмування в STEP 7 MicroWIN

При написанні програми керування необхідно керуватися принциповими схемами наведеними у «Альбомі схем».

Нижче наведені інструкції з написання програми управління транспортером.

Прив'яжемо роботу приводу транспортера до органів керування (наприклад по натисканню кнопки). Нехай поки натиснута кнопка SB1 транспортер працює, коли кнопка відпущена транспортер не працює. Реалізація цієї програми представлена на рис. 1.17 (згідно альбому SB1підімкнений до I0.2).

Необхідно набрати цю програму в STEP 7 Micro/WIN

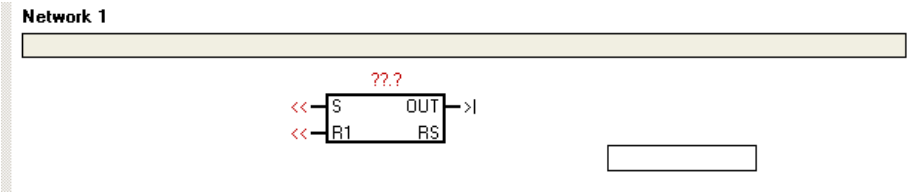


Рисунок 1.17 – Введення команди RS

(Інколи при установці елементів можуть з'являтися знаки «??.?». В таких випадках елементам необхідно присвоювати адреси області пам'яті.)

На вхід тригера *S* встановлюємо кнопку *SB1* за допомогою вкладки Select symbol з контекстного меню.

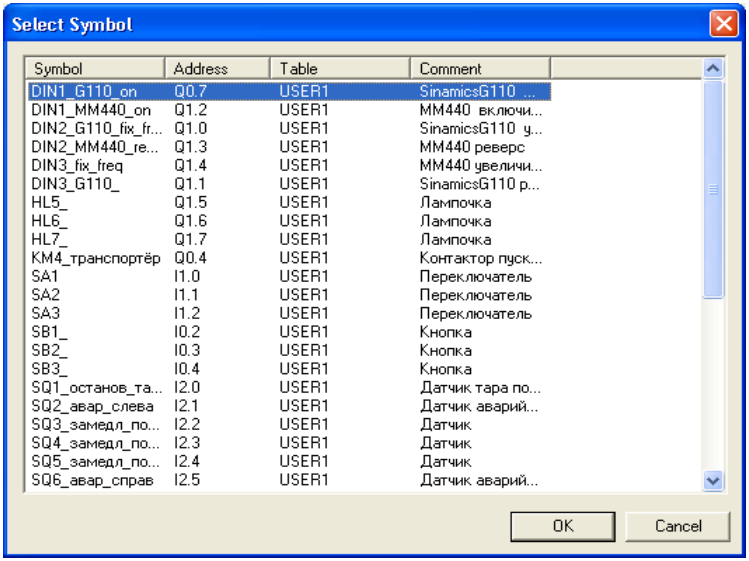
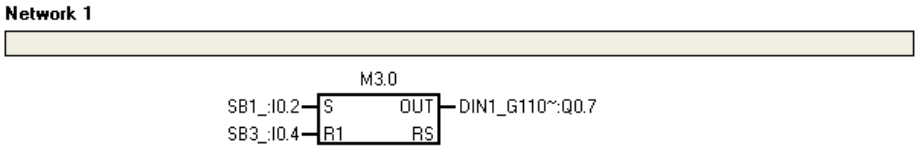


Рисунок 1.18 – Вікно Select symbol для вибору сигналу за символьним іменем

На вхід тригера *R* встановлюємо кнопку *SB3*.

Вихідний сигнал з тригера будемо подавати на вхід приводу Sinamics G110.



Symbol	Address	Comment
DIN1_G110_on	Q0.7	SinamicsG110 включить/выключить
SB1_	I0.2	Кнопка
SB3_	I0.4	Кнопка

Рисунок 1.19 – Готова програма

Запускаємо модель LiquidDozing.mdl, вводимо на екран візуалізації моделі (рис.120).

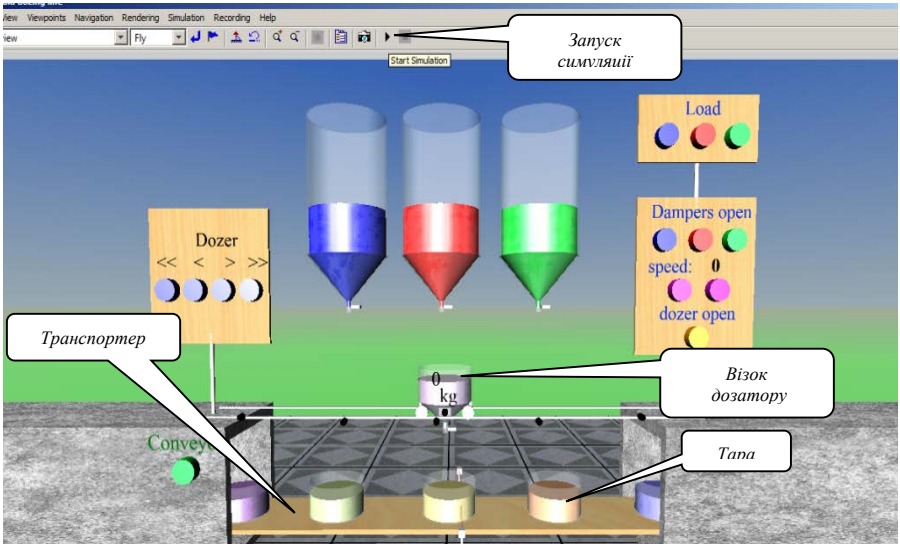


Рисунок 1.20 – Модель LiquidDozing

Запускаємо STEP 7 MicroWIN в режимі онлайн**,** і запускаємо програму в режимі Run.

Для перевірки натискаємо і утримуємо кнопку SB1, і спостерігаємо на візуалізації моделі пересування конвеєра з тарою.

1.17.3 Програмне забезпечення необхідне для роботи зі стендом

Для програмування програмованого контролера SIMATIC S7-200 (CPU226) використовується програма STEP 7 MicroWIN V3.2 фірми SIEMENS. Ця програма дозволяє написати програму мовою STEP7 і записати її в контролер. Крім того, можна переглянути в режимі он-лайн правильність роботи програми та налагодити її. Комп'ютер приєднується до SIMATIC S7-200 (CPU226) на стенді через кабель програмування PPI.

Сенсорну панель TP170micro також програмується через кабель PC / PPI. Сама програма візуалізації пишеться за допомогою середовища розробки WinCC Flexible2005micro. Ця програма також дозволяє налагодити візуалізацію у режимі симуляції.

Для параметрування приводів SINAMICS G110 і MICROMASTER 440 використовується програма STARTER фірми SIEMENS. Вона дозволяє через PC інтерфейсну плату приводів з допомогою інтерфейсного кабелю підключеного до COM-порту (Sinamics G110 підключений до COM1, а MicroMaster440 підключений до COM2) налаштувати ці приводи на всі можливі для даного приводу режими і настройки. Також ця програма дозволяє в он-лайн режимі подивитися поточні параметри і стан приводу і електродвигуна.

1.18 Завдання

Необхідно при встановленні запиту на «зміну тари» (згідно Symbol table ….), привести в рух транспортер. При спрацюванні давача SQ1, транспортер повинен зупинитися. Відповідно встановиться біт, це означатиме що транспортер знаходиться на позиції «Тара на позиції зливу» (згідно Symbol table ….), після цього запит скидається.

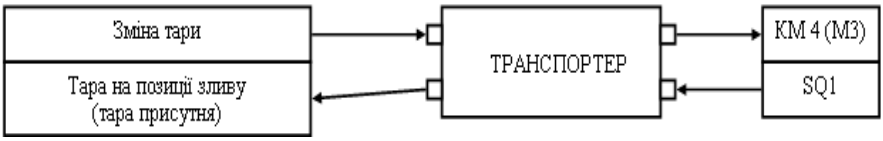


Рисунок 1.21 – Блок-схема програми роботи транспортера

1.19 Контрольні питання

1. Для чого в даній роботі використовуємо самоблокуюче реле.
2. Перерахувати основні логічні функції STEP 7 Micro/WIN.
3. Розповісти, як зберігається інформація до даних в S7-200.
4. Назвати та розповісти про області пам'яті даних.
5. Розповісти про спеціальні біти пам'яті.
6. Розповісти про команди котушок STEP 7 Micro/WIN.

2 ЛАБОРАТОРНА РОБОТА №2

КЕРУВАННЯ ПОЗИЦІОНУВАННЯМ ВІЗКА ДОЗАТОРА

Мета: ознайомитися з пристроєм і функціональними можливостями стенду засобів автоматизації фірми SIEMENS. Оцінити обсяг робіт, які належить виконати в цьому курсі на цьому стенді. Зрозуміти палітру пристроїв автоматизації та перспективи використання цього стенду в інших курсах, в дипломному проектуванні, а також у застосуванні отриманих на ньому навичок для вирішення завдань автоматизації і для модернізації існуючих систем керування.

Розібратися в управлінні системи позиціонування тари, навчитися керувати рухом тари при різних завданнях.

2.1 Короткі теоретичні відомості

У даній лабораторній роботі будуть використовуватися кілька нових функцій, ще також буде використовуватися теорія минулих лабораторних робіт.

Нижче наведені деякі спеціальні функції (SF).

2.2 Таймери: T

S7-200 має в своєму розпорядженні таймери, які відраховують збільшення часу з дозволами (кроками бази часу) 1 мс, 10 мс або 100 мс. З таймером пов'язані дві змінні:

1:поточне значення: це 16-бітове ціле зі знаком зберігає кількість часу, відлічених таймером.

2:біт таймера: цей біт встановлюється або скидається, коли поточне значення стає рівним передвстановленому значенню.

Передустановлене значення вводиться як частина таймерної команди.

Ви звертаєтеся до обох цих елементів даних через адресу таймера (T + номер таймера). Чи відбувається звернення до біту таймера або до поточного значення, залежить від використовуваної команди: команди з операндами в бітовому форматі звертаються до біту таймера, тоді як команди з операндами 8 форматі слова звертаються до поточного значення. Як показано на рис. 4-5, команда "Нормально відкритий контакт" звертається до біту таймера, а команда "Передати слово" звертається до поточного значення таймера.

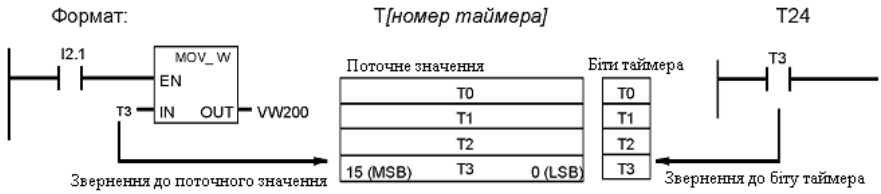


Рисунок 2.1 – Звернення до біта чи до поточного значення таймера.

2.2 Логічні операції

2.2.1 Операції інвертування

Інвертування байта, слова та подвійного слова.
Команди інвертування байта (INVB), слова (INVW) і подвійного слова (INVD) утворюють додаток входу IN до одиниці і завантажують результат за адресою OUT.

Збійні стану, що встановлюють ENO = O

■ 0006 (непряма адреса)

Біти спеціальної пам'яті, на які діє команда:

■ SM1.0 (нуль)

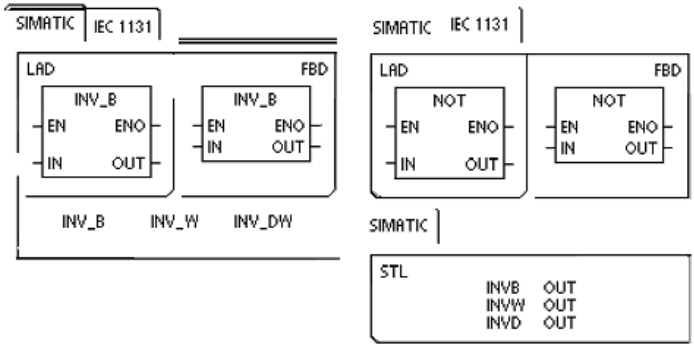
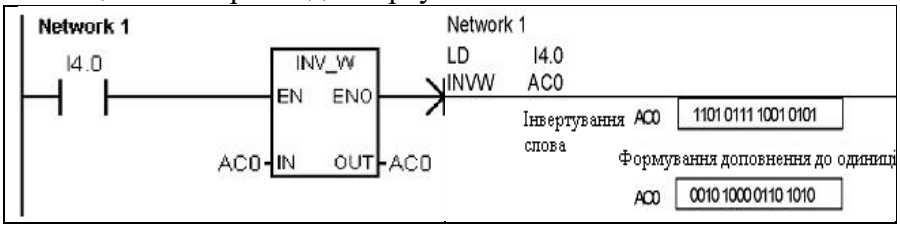


Рисунок 2.2 - Операції інвертування

Таблиця 2.1 – Допустимі операнди для команд інвертування

Входи / виходи	Типи даних	Операнди
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, * AC, константа
	DWORD	ID, OD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD. *LD, *AC
	WORD	IW, OW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Таблиця 2.2 – Приклад інвертування



2.2.2 Порозрядні логічні операції І, АБО і виключне АБО

Порозрядне логічне І з байтами, словами і подвійними словами.

Порозрядні логічні операції І з байтами (ANDB), словами (ANDW) і подвійними словами (ANDD) логічно сполучають відповідні біти двох вхідних величин INI і IN2 згідно з таблицею істинності логічної операції І і завантажують результат за адресою OUT.

Порозрядне логічне АБО з байтами, словами і подвійними словами.

Порозрядні логічні операції АБО з байтами (ORB), словами (ORW) і подвійними словами (ORD) логічно сполучають відповідні біти двох вхідних величин IN1 і IN2 згідно з таблицею істинності логічної операції АБО і завантажують результат за адресою OUT.

Порозрядне логічне виключне АБО з байтами, словами і подвійними словами.

Порозрядні логічні операції виключає АБО з байтами (XORB), словами (XORW) і подвійними словами (XORD) логічно сполучають відповідні біти двох вхідних величин IN1 і IN2 згідно з таблицею істинності логічної операції виключає АБО і завантажують результат за адресою OUT.

Біти спеціальної пам'яті і ENO

Для всіх команд, описаних на цій сторінці, такі умови впливають на біти спеціальної пам'яті і ENO.

Збійні стани, встановлюючи ENO = О

■ 0006 (непряма адреса)

Біти спеціальної пам'яті, на які діє команда:

■ SM 1.0 (нуль)

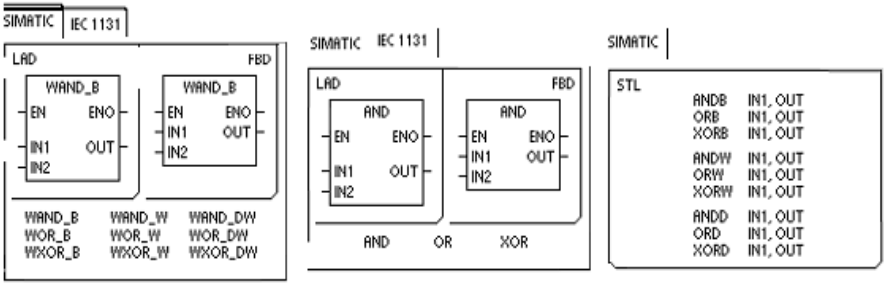
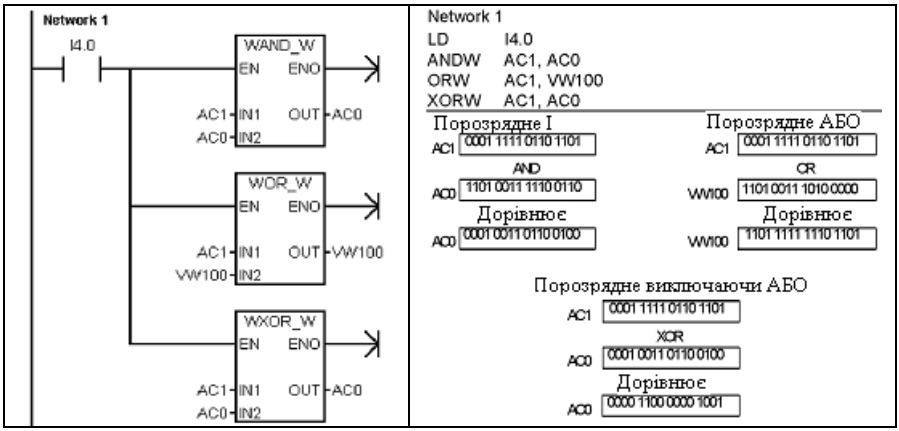


Рисунок 2.3 – Біти спеціальної пам'яті і ENO

Таблиця 2.3 – Допустимі операнди для операцій І, АБО і виключаюче АБО.

Входи / виходи	Типи даних	Операнди
IN1; IN2	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, * AC, константа
	DWORD	ID, OD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD. *LD, *AC
	WORD	IW, OW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Таблиця 2.4 – Приклад: Порозрядні логічні операції І, АБО і виключне АБО



2.3 Команди керування програмою

2.3.1 Умовне завершення

Команда умовного завершення (END) завершує поточний цикл в залежності від результату попередньої логічної операції. Ви можете використовувати команду умовного завершення у головній програмі, але її не можна використовувати в підпрограмі і програмах обробки переривань.

2.3.2 Зупинка

Команда зупинки (STOP) завершує виконання програми, викликаючи перехід CPU S7-200 з RUN в STOP.

Якщо команда STOP виконується в програмі обробки переривання, то ця програма завершується негайно, а всі переривання, що стоять в черзі, ігноруються. Решта дії в поточному циклі обробки програми завершується, включаючи виконання головної програми користувача, а перехід з RUN в STOP проводиться в кінці поточного циклу.

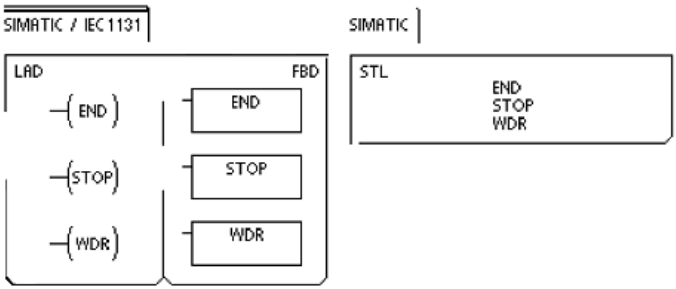


Рисунок 2.4 – Умовне завершення, зупинка.

2.3.3 Скидання контролю часу

Команда скидання контролю часу (WDR) перезапускає системний таймер контролю часу CPU S7-200. збільшуючи час, який може займати цикл обробки програми, не викликаючи помилки контролю часу.

Команду скидання контролю часу слід використовувати з обережністю. Якщо ви за допомогою програмних циклів перешкоджаєте завершення циклу обробки програми або істотно затримуєте його завершення, то слід мати на увазі, що до завершення циклу обробки програми заборонені наступні процеси:

- зв'язок (за винятком режиму вільно програмованого обміну даними Freeport);
- оновлення входів і виходів (крім входів і виходів з безпосереднім доступом);
- оновлення, примусово задаються значення;
- оновлення бітів спеціальної пам'яті (не оновлюються біти SMO, SM5 - SM29);
- діагностика в режимі реального часу;
- 10-мілісекунд і 100-мілісекунд таймери не накопичують час належним чином для циклів обробки програми, які перевищують 25 секунд;

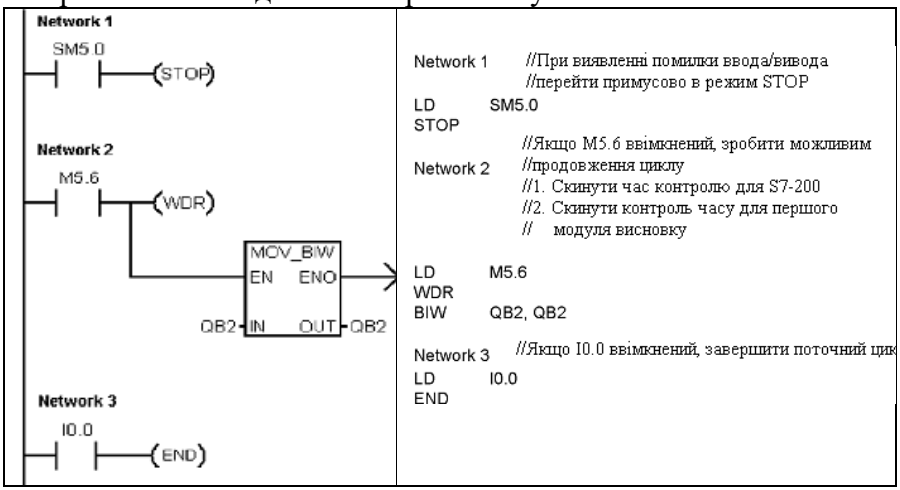
- команда STOP при використанні в програмі обробки переривання;
- модулі розширення з цифровими виходами також мають таймер контролю часу, який вимикає виходи, якщо S7-200 не робить запис в цей модуль. Щоб ці виходи залишалися включеними під час продовженого циклу, користуйтеся командою безпосереднього запису для кожного модуля розширення. Розглянемо також приклад, наступний за цим описом.

Порада! Команду скидання контролю часу слід використовувати для перезапуску таймера контролю часу, якщо ви очікуєте, що час циклу обробки програми перевищить 500 мс, або ви очікуєте збільшення активності переривань, що може перешкодити поверненню в головний цикл більш ніж на 500 мс.

Кожен раз, коли ви застосовуєте команду скидання контролю часу, ви повинні також використовувати команду безпосереднього запису для одного вихідного байта (QB) на кожен цифровий модуль виводу, щоб скинути контроль часу модуля розширення.

Якщо ви використовуєте команду скидання контролю часу, щоб можна було виконати програму з великим часом циклу, то переклад перемикача режимів роботи в положення STOP викликає перехід S7-200 в стан STOP протягом 1,4 секунди.

Таблиця 2.5 – Приклад: Команда зупинки, умовне завершення та скидання контролю часу.



2.4 Команди формування програмного циклу For-Next

За допомогою команд FOR і NEXT ви можете керувати програмними циклами, які повторюються декілька разів. Кожна команда FOR вимагає наявності команди NEXT. Ви можете вкладати цикли FOR / NEXT один в одного (поміщати цикл FOR-NEXT всередині іншого циклу FOR-NEXT). Глибина вкладення не може перевищувати вісім.

Команда FOR виконує команди, розташовані між операторами FOR і NEXT. Ви повинні задати значення індексу або лічильник циклу INDX, початкове значення IN IT і кінцеве значення FINAL.

Команда NEXT відзначає кінець циклу FOR.

Збійні стани, що встановлюють ENO = O

■ 0006 (непряма адреса)

Якщо ви активізуєте цикл FOR-NEXT, то процес циклічного виконання продовжується, поки не закінчаться ітерації, якщо тільки ви не зміните кінцеве значення зсередини

самого циклу. Ви можете змінювати ці значення, поки цикл FOR-NEXT виконає циклічну обробку. Коли цикл активізується знову, він копіює початкове значення в індекс (лічильник циклу).

Команда FOR-NEXT скидає себе кожен раз, коли вона активізується.

Наприклад, якщо значення INIT дорівнює 1, значення FINAL дорівнює 10. то команди між FOR і NEXT виконуються 10 разів, причому значення INDX щоразу збільшується на одиницю: 1,2, 3, ... 10.

Якщо початкове значення більше кінцевого, то цикл не виконується. Після кожного виконання команд між FOR і NEXT значення INDX збільшується, а результат порівнюється з кінцевим значенням. Якщо INDX більше кінцевого значення, то цикл завершується.

Якщо вершина стека дорівнює 1, коли ваша програма входить до циклу FOR-NEXT то вершина стека залишиться рівною 1, коли ваша програма покине цикл FOR-NEXT.

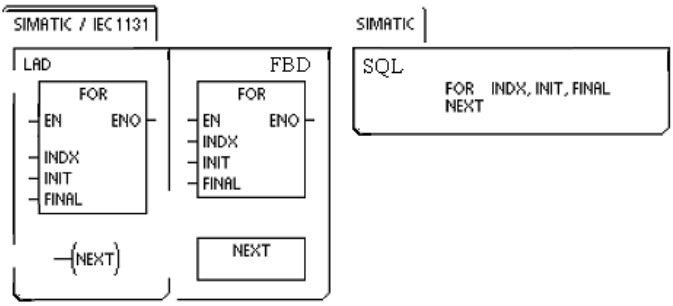
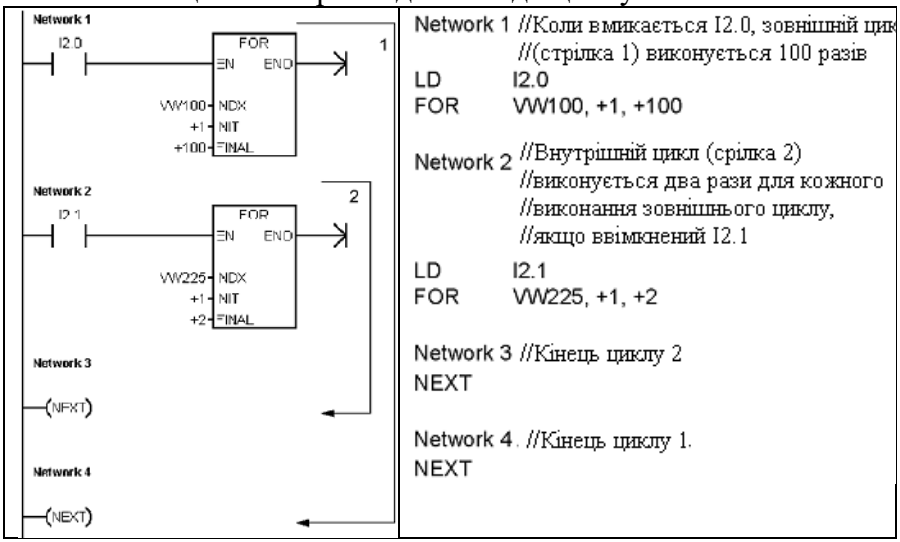


Рисунок 2.5 – Команди формування програмного циклу For-Next

Таблиця 2.6 – Допустимі операнди для команди For і Next

Входи / виходи	Типи даних	Операнди
INDX	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
INIT FINAL	INT	VW, QW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC константа

Таблиця 2.7 – Приклад: команди циклу For і Next



2.5 Команди переходу

Команда переходу на мітку (JMP) здійснює перехід до зазначеної мітки N всередині програми:

Команда "Позначка" (LBL) відзначає становище мети переходу N.

Команду переходу на мітку можна використовувати в основній програмі, в підпрограмі і в програмах обробки переривань. Команда переходу і відповідна мітка завжди повинні

знаходитися всередині одного і того ж сегменту коду (в основній програмі, підпрограмі або програмі обробки переривань).

Ви не можете перейти з головної програми на мітку в підпрограмі або в програмі обробки переривання. Аналогічно, ви не можете перейти з підпрограми або програми обробки переривання на мітку поза цією підпрограмою, або програму обробки переривання.

Команду переходу на мітку можна використовувати всередині сегмента SCR, але відповідна позначка повинна знаходитися всередині того ж сегменту SCR.

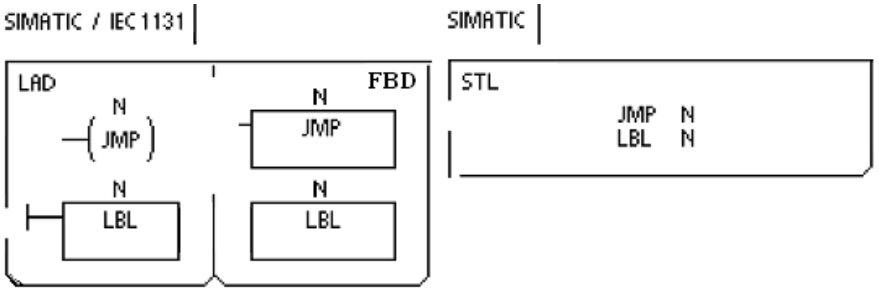
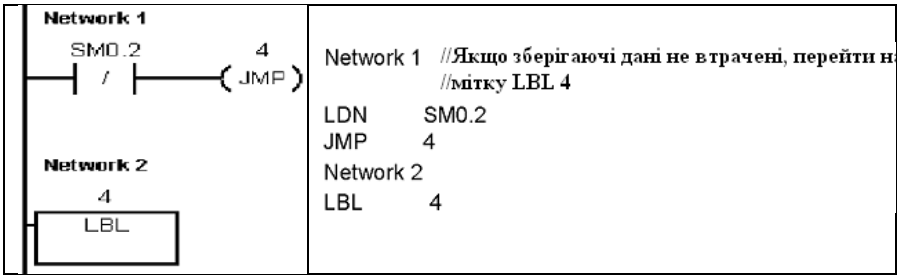


Рисунок 2.6 – Команди переходу

Таблиця 2.8 – Допустимі операнди для команд переходу

Входи / виходи	**Типи даних**	**Операнди**
N	WORD	Константа (від 0 до 255)

Таблиця 2.9 – Приклад: Команди переходу на мітку



2.6 Команди зсуву і циклічного зсуву

2.6.1 Команди зсуву вправо і зсуву вліво

Команди зсуву зрушують вхідну величину IN вправо або вліво на число розрядів, вказане в N (і завантажують результат у вихід OUT).

Команди зсуву заповнюють позиції висунутих бітів нулями. Якщо величина зсуву (N) більше або дорівнює максимально допустимому значенню (8 для операцій з байтами, 16 для операцій зі словами і 32 для операцій з подвійними словами), то зсув виробляється на максимально можливу величину для даної операції. Якщо величина зсуву більше 0, то біт переповнення (SM1.1) приймає значення останнього висунутого біта. Біт нульового значення (SM1.0) встановлюється, якщо результат операції зсуву дорівнює нулю.

Байтові операції є беззнаковими. Для операцій із словами і подвійними словами знаковий біт зсувається, якщо ви використовуєте типи даних зі знаком.

Збійні стану, що встановлює ENO = 0

■ 0006 (непряма адреса)

Біти спеціальній пам'яті, на які діє команда:

■ SM1.0 (нуль)

■ SM1.1(переповнення)

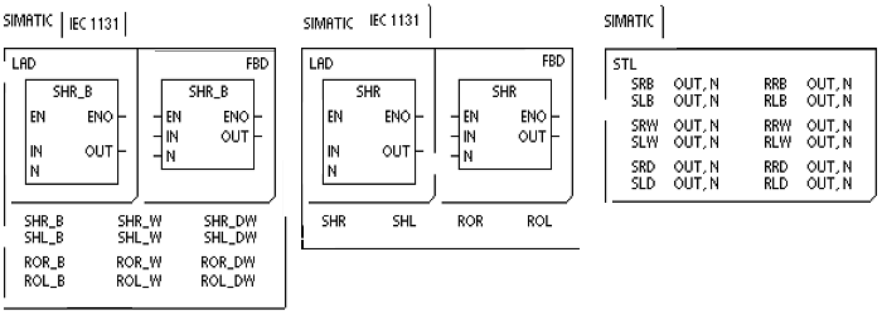


Рисунок 2.7 – Команди зсуву вправо і зсуву вліво

2.6.2 Команди циклічного зсуву вправо і циклічного зсуву вліво

Команди циклічного зсуву циклічно зрушують вхідну величину (IN) вправо або вліво на число розрядів, вказане в (N). і завантажують результат за адресою (OUT). Циклічний зсув є кільцевим.

Якщо величина зсуву більше або дорівнює максимально допустимому значенню (8 для операцій з байтами, 16 для операцій зі словами і 32 для операцій з подвійними словами), то S7-200 виконує операцію по модулю з величиною зсуву (одержання залишку від ділення заданого зсуву на максимально допустимий), щоб отримати дійсну величину циклічного зсуву. Результатом є величина зсуву від 0 до 7 для операцій з байтами, від 0 до 15 для операцій зі словами і від 0 до 31 для операцій з подвійними словами.

Якщо величина зсуву дорівнює нулю, то циклічний зсув не проводиться. Якщо циклічний зсув виконується, то значення останнього циклічно зрушеного біта копіюється в біт переповнення (SM1.1).

Якщо величина зсуву не є цілим кратним 8 (для операцій з байтами), 16 (для операцій зі словами) або 32 (для операцій з подвійними словами), то останній циклічно висунутий біт копіюється в біт переповнення (SM1.1). Біт нульового значення

(SM1.0) встановлюється, якщо підлягає циклічному зсуву величина дорівнює нулю.

Операції з байтами є беззнаковими. Для операцій із словами і подвійними словами знаковий біт зсувається, якщо ви використовуєте типи даних зі знаком.

Збійні стану, що встановлює ENO = 0

■ 0006 (непряма адреса)

Біти спеціальній пам'яті, на які діє команда:

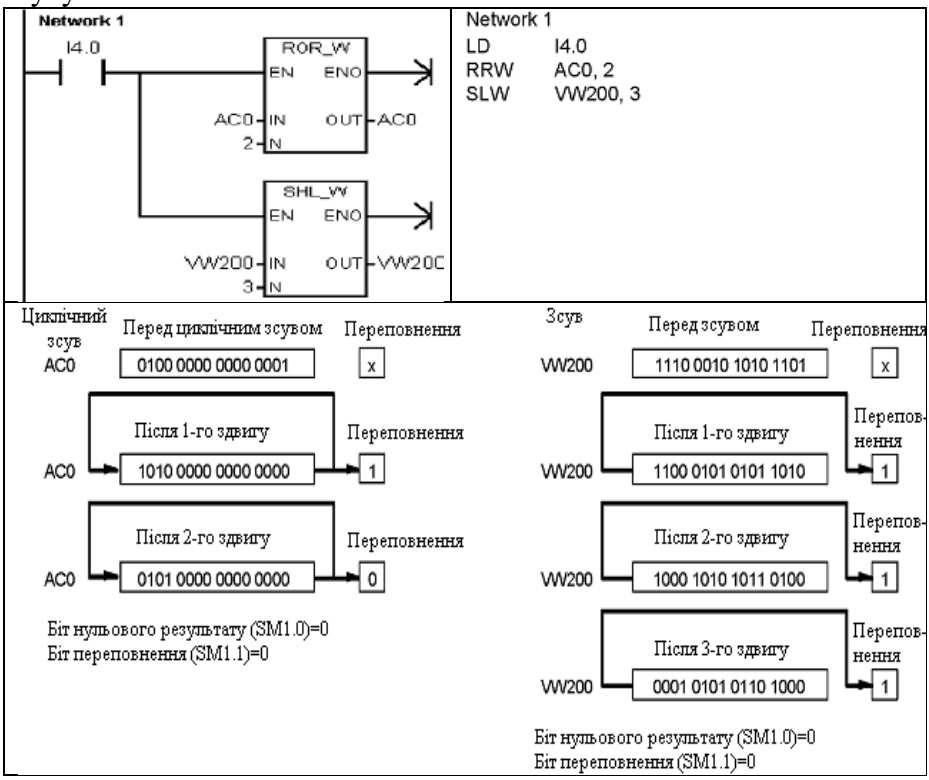
■ SM1.0 (нуль)

■ SM1.1(переповнення)

Таблиця 2.10 – Допустимі операнди для команд зсуву і циклічного зсуву.

Входи / виходи	**Типи даних**	**Операнди**
IN	BYTE	IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC константа
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE	IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC
N	BYTE	IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC константа

Таблиця 2.11 – Приклад: Команди зсуву і циклічного зсуву



2.7 Операції з підпрограмами

Команда виклику підпрограми (CALL) передає управління підпрограмі SBR_N. Команду виклику підпрограми можна використовувати з параметрами, або без них. Як тільки виконання підпрограми завершується, управління повертається команді, наступної за викликом підпрограми.

Команда умовного повернення з підпрограми (CRET) завершує підпрограму в залежності від результату попередньої логічної операції.

Щоб додати підпрограми виберіть команду меню Edit -»Insert -> Subroutine [Редагувати -> Вставити -> Підпрограма]

Збійні стану, що встановлюють ENO = O

■ 0008 (перевищена максимальна вкладеність для підпрограм)

■ 0006 (непрямий адреса)

У головній програмі ви можете вкладати підпрограми один в одного (поміщати виклик підпрограми усередині іншої підпрограми) на глибину до восьми рівнів. У програмі обробки переривання вкладення підпрограм один в одного неможливо.

Виклик підпрограми не може бути поміщений в жодну іншу підпрограму, що спричинюється з програми обробки переривання. Рекурсія (виклик підпрограми, що викликає саму себе) не заборонена, але з підпрограмами її слід використовувати з обережністю.

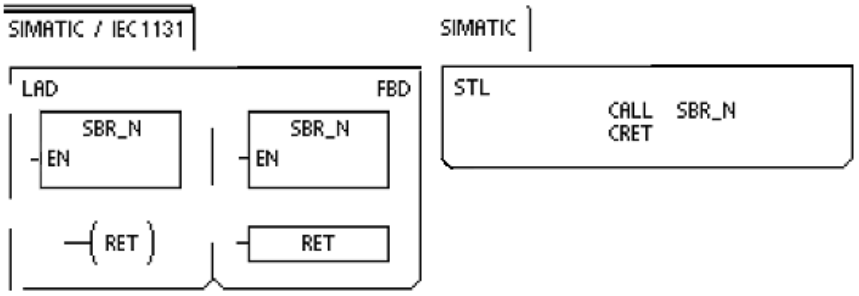


Рисунок 2.8 – Операції з підпрограмами

Таблиця 2.12 – Допустимі операнди для команди виклику підпрограми

Входи / виходи	Типи даних	Операнди
SBR_N	WORD	константа для CPI 221. CPI 222. CPI 224: від 0 до 63 для CPI 224XP і CPI 226 від 0 до 127
IN	BOOL BYTE WORD, INT DWORD,DINT STRING	V, I Q, M, SM, S, T, C, L потік сигналу VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC[1], константа VW, T, C,IW, QW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC[1], константа VD, ID, QD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC[1], &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ,, &SMB константа VD, LD, AC, константа
IN / OUT	BOOL BYTE WORD, INT DWORD,DINT	V, I Q, M, SM[2], S, T, C, L VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1] VW, T, C,IW, QW, MW, SMW[2], SW, LW, AC, AIW, *VD, *LD, *AC[1] VD, ID, QD, MD, SMD[2], SD, LD, AC, HC, *VD, *LD, *AC[1]
OUT	BOOL BYTE WORD, INT DWORD,DINT	V, I Q, M, SM[2], S, T, C, L VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1] VW, T, C,IW, QW, MW, SMW[2], SW, LW, AC, AIW, *VD, *LD, *AC[1] VD, ID, QD, MD, SMD[2], SD, LD, AC, HC, *VD, *LD, *AC[1]

[1] Зсув повинен бути не менше 1

Порада! STEP 7-Micro/WIN автоматично додає команду безумовного повернення з кожної підпрограми.

Коли викликається підпрограма, весь логічний стек зберігається, вершина стека встановлюється в одиницю, всі інші осередки стека встановлюються в нуль і керування передається викликанню підпрограми. Коли ця підпрограма завершується, стек відновлюється зі значеннями, збереженими в точці виклику, а управління повертається в зухвалу програму.

Акумулятори є загальними для підпрограм і викликає програми. При використанні підпрограми операції збереження і відновлення до акумуляторів не застосовуються.

Якщо підпрограма викликається в одному і тому ж циклі кілька разів, то не можна застосовувати команди «Наростаючий фронт», «Падаючий фронт», а також таймери і лічильники.

2.7.1 Виклик підпрограми з параметрами

Підпрограма може містити передані параметри. Параметри визначаються в таблиці локальних змінних підпрограми. Параметру повинно бути призначено символічне ім'я (не більше 23 символів), тип змінної та тип даних. У підпрограму і з неї може бути передано шістнадцять параметрів.

Поле типу змінної в таблиці локальних змінних визначає, чи передається змінна в підпрограму (IN), в підпрограму і з її (IN_OUT), або вона передається з підпрограми (OUT) Типи параметрів для підпрограми описані в таблиці 2.13 Для додавання параметра помістіть курсор на полі того типу параметрів (IN, IN_OUT або OUT), який ви хочете додати. Клацніть правою кнопкою миші, щоб викликати меню для вибору. Виберіть пункт Insert [Вставити], а потім пункт Row Below [Рядок знизу] Під поточним записом з'явиться місце для запису ще одного параметра вибраного типу.

Таблиця 2.13 – Типи параметрів для підпрограми

Параметр	Опис
IN	Параметри передаються в підпрограму. Якщо параметр є прямою адресою (Наприклад, VBI0). то в підпрограму передається значення, що знаходиться за вказаною адресою. Якщо параметр є непрямою адресою (наприклад, * AC1), то в підпрограму передається значення, що знаходиться за адресою, на який зроблено зсипання. Якщо параметр є константою (16 # 1234) або адресою (& VB100). то в підпрограму передається значення константи або адреси.
IN_OUT	Значення, що знаходиться за вказаною адресою параметра, передається в підпрограму, а результуюче значення повертається за тією ж самою адресою. Константи (наприклад, 16 # 1234) і адреси (наприклад, SVBI00) не можуть бути параметрами типу IN_OUT.
OUT	Результуюче значення з підпрограми повертається за вказаною адресою параметра. Константи (наприклад. 16 # 1234) та адреси (наприклад, & VB100) не можуть бути параметрами типу OUT. Так як вихідні параметри не зберігають значення, присвоєного останнім виконанням підпрограми то ви повинні присвоювати значення виходів при кожному виклику підпрограми. Зверніть увагу, що команди SET і RESET впливають на значення булевих операндів тільки в тому випадку, якщо потік сигналу включений ON.
TEMP	Локальна пам'ять, не використовувана для переданих параметрів, може використовуватися

	для тимчасового зберігання даних усередині підпрограми.

Як показано на рис. 2.9, поле типу даних в таблиці локальних змінних визначає розмір і формат параметра. Нижче перераховані типи параметрів:

- BOOL; Цей тип даних використовується для окремих бітових входів і виходів. IN3 в наступному прикладі є булевим входом. – BYTE, WORD, DWORD: Ці типи даних визначають вхідний або вихідний параметр без знаку розміром 1, 2 або 4 байти відповідно.
-

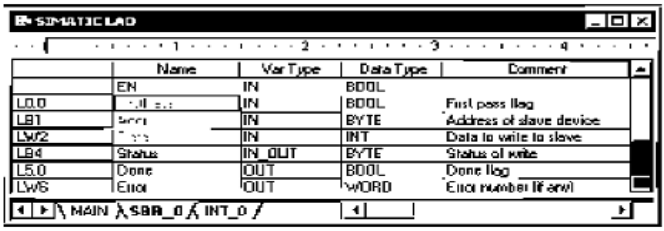


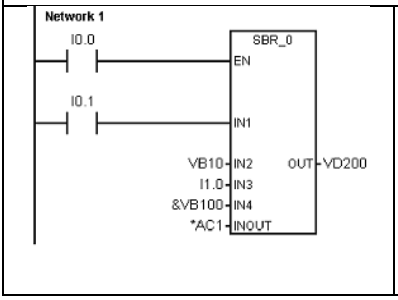
Рисунок 2.9 – Таблиця локальних змінних

- BYTE, WORD, DWORD: Ці типи даних визначають вхідний або вихідний параметр без знаку розміром 1, 2 або 4 байти відповідно.
- INT, DINT: Ці типи даних визначають вхідний або вихідний параметр без знаку розміром 1, 2 або 4 байти відповідно
- REAL: Цей тип даних визначає число з плаваючою точкою IEEE одноразової точності (4 байти).
- STRING: Цей тип даних використовується в якості чотирьохбайтового покажчика на рядок.
- Power Flow [Потік сигналу): Логічний потік сигналу дозволений тільки для бітових (булевих) входів. Цей опис повідомляє STEP 7-Micro/WIN 32, що цей вхідний параметр є результатом досягнення підпрограми потоком

сигналу, заснованим на комбінації бітових логічних операцій. Входи з булевим потоком сигналу повинні знаходитися в таблиці локальних змінних перед будь-яким іншим типом входів. Таким способом можна використовувати тільки вхідні параметри. Дозвільний вхід (EN) і вхід IN1 в наступному прикладі використовують булеву логіку.

Таблиця 2.14 – Приклад: Виклик підпрограми

Нижче наведено два приклади на STL. Перший набір команд STL може бути відображений тільки в редакторі STL, так як булеві параметри, використовувані як входів типу "Потік сигналу", не зберігаються в локальній пам'яті.
Другий набір команд STL може бути відображений також і у редакторах LAD і FBD. так як для збереження станів булевих вхідних параметрів, які в LAD і FBD показані як входи, що приймають потік сигналу, використовується локальна пам'ять.



Тільки STL:

```
Network 1
LD      I0.0
CALL    SBR_0, I0.1, VB10, I1.0, &VB100, *AC1, VD200
```

Для правильного відображенняв LAB і FBD

```
Network 1
LD      I0.0
=       L60.0
LD      I0.1
=       L63.7
LD      L60.0
CALL    SBR_0, L63.7, VB10, I1.0, &VB100, *AC1,
VD200
```

Адресні параметри, наприклад, IN4 (& VB100) передаються в підпрограму як DWORD (подвійне слово без знака). Тип постійного параметра повинен бути зазначений для параметра в зухвалій програмі за допомогою описувача константи перед значенням константи. Наприклад, щоб передати в якості параметра константу, що має розмір подвійного слова без знака, зі значенням 12 345, постійний параметр повинен бути

заданий як DW # 12345. Якщо описувач константи для параметра опущений, то константа може бути сприйнята як має інший тип.

Автоматичне перетворення типів для вхідних і вихідних параметрів не проводиться. Наприклад, якщо таблиця локальних змінних вказує, що параметр має тип даних REAL, а викликана програма задає для цього параметра подвійне слово (DWORD), то це значення в підпрограмі буде розглядатися як подвійне слово.

Коли значення передаються в підпрограму, вони поміщаються в локальну пам'ять підпрограми. Самий лівий стовпець таблиці локальних змінних показує адресу в локальній пам'яті для кожного переданого параметра. Значення вхідних параметрів копіюються в локальну пам'ять підпрограми, коли підпрограма викликається. Значення вихідних параметрів копіюються з локальної пам'яті підпрограми в зазначені адреси вихідних параметрів, коли виконання підпрограми завершується.

Розмір і тип елемента даних представляються в коді параметра. Значення параметрів ставляться, а відповідність локальної пам'яті в підпрограмі наступним чином:

- Значення параметрів ставляться у відповідність локальної пам'яті в порядку, який задається командою виклику підпрограми з параметрами, починаючи з LO.
- Від одного до восьми послідовних бітових значень параметрів ставляться у відповідність окремим байту, починаючи з Lx.O і аж до Lx.7.
- Значення, що мають тип байт, слово або подвійне слово ставляться у відповідність локальної пам'яті на кордонах байтів (LBx, LWx або LDx).

У команді виклику підпрограми з параметрами, параметри повинні бути розташовані в такому порядку: спочатку вхідні параметри, за ними параметри типу IN_OUT, а потім вихідні параметри.

Якщо ви програмуєте на STL. то формат команди CALL має вигляд:

CALL номер підпрограми, параметр 1. параметр 2, ... , параметр п

2.8 Опис роботи візка та датчиків положення

Рух візка представлено на рис. 2.10. Для позиціонування візка використовуються датчики положення SQ2-SQ7. Датчики SQ2, SQ6 – кінцеві вимикачі (нормально-замкнуті контакти).
Датчик SQ7 встановлений на візку, він служить для точної зупинки, і спрацьовує по вузькому прапору, над точками завантаження.

SQ3-SQ5 (нормально-розімкнуті контакти) – використовуються для уповільнення швидкості з метою більш точного позиціонування візка під завантаження. Давачі розташовані під своїм бункером з фарбою відповідно: SQ3 – синій; SQ4 – червоний; SQ5 – зелений.

Давач спрацьовує з детектування смуги залізного куточка встановленого на візку.

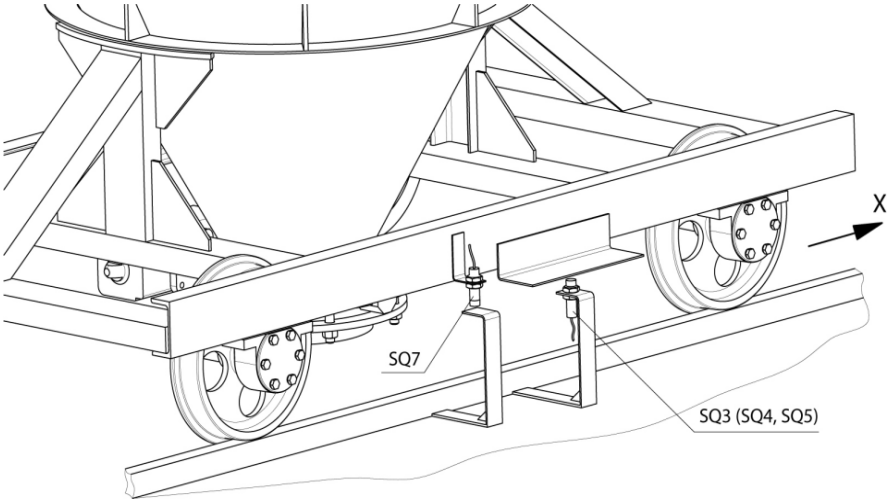


Рисунок 2.10 – Загальний вигляд візка і елементів системи позиціонування

Діаграма станів давачів представлено на рис. 2.11.

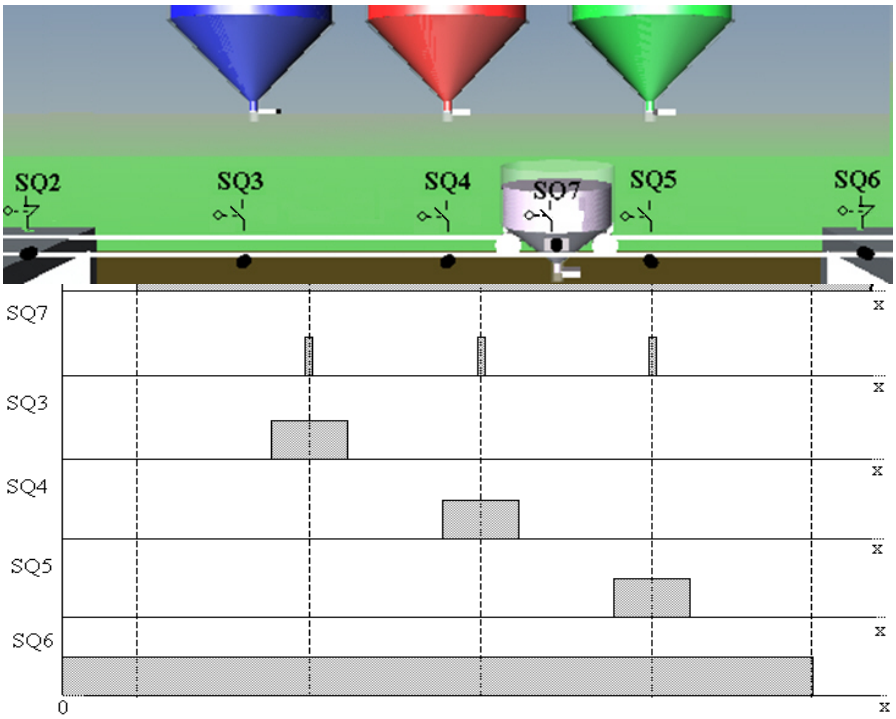


Рисунок 2.11 – Діаграма стану давачів положення від розташування візка

2.9 Хід виконання лабораторної роботи

Для виконання даної лабораторної роботи необхідно виконати наступні дії:

1. Перекомутувати комутатор, налаштувавши на дозування (за допомогою програми AccessPort: D \ DA_stend \ connect \ _AccessPort).

2. Відкрити модель дозування (на комп'ютері для моделювання: D \ DA_stend \ models \ 2_LiquidDozing \ LiquidDozing.mdl).

3. Відкрити програму STEP 7 Micro/WIN, для програмування конвеєра (на комп'ютері для програмування).

4. Налаштувати програму STARTER.

2.10 Налаштування стенду і комп'ютерів на роботу

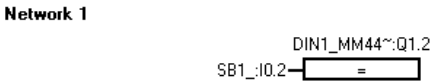
Налаштування Access Port, і програми STARTER, виконується аналогічно раніше виконаних лабораторних робіт**.**

2.11 Програмування в STEP 7 Micro/WIN

При написанні програми керування необхідно керуватися принциповими схемами наведеними в «Альбомі схем».

Реалізуємо рух візка при натисканні кнопки SB1.При відпусканні кнопки візок зупиняється. Щоб усунути цей недолік у схему додаємо елемент пам'яті, наприклад тригер. Для зупинки візка у потрібному місці на вихід тригера R необхідно подати сигнал з давачів (для переходу на знижену швидкість і точної зупинки).

Приклад готової програми показаний на рис. 2.12



Symbol	Address	Comment
DIN1_MM440_on	Q1.2	MM440 включить/выключить
SB1_	I0.2	Кнопка

Рисунок 2.12 - Приклад готової програми

2.12 Завдання

1. При натисканні на кнопку SB1 візок дозатора повинен їхати вліво, до наступної точки зупину під завантаженням (по спрацьовуванню давача SQ7 спільно з давачами уповільнення) або наїзду на кінцевий вимикач SQ2, за аналогією рух вправо натисканням кнопки SB3. Натискання кнопки SB2 зупиняє рух візка дозатору.

2. Забезпечити зміну швидкості пересування візка дозатора (зменшити до 15 Гц), коли вона знаходиться не в зоні дії давачів SQ3, SQ4, SQ5.

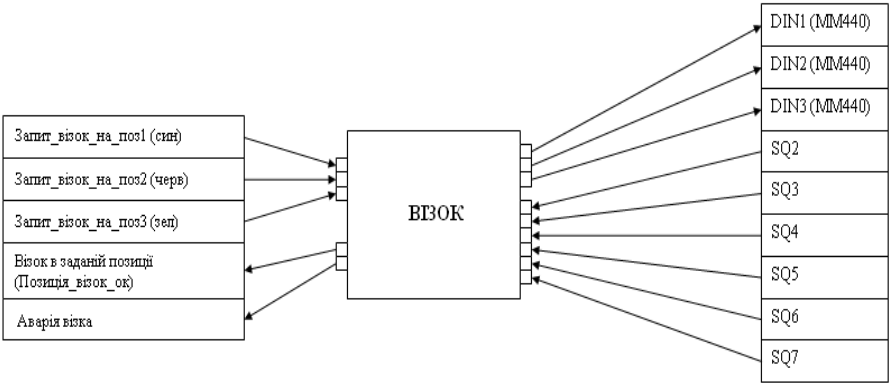


Рисунок 2.13 – Блок-схема програми роботи візка

2.13 Контрольні питання

1. Розповісти про принцип роботи таймеру
2.Назвати та розповісти про логічні операції, які використовуються в даній роботі.
3. Розповісти про команди керування програмою.
4. Розповісти про команди формування програмного циклу For-Next.
5. Розповісти про команди зсуву і циклічного зсуву.
6. Покажіть у програмі STARTER, де потрібно міняти частоту, щоб візок прискорювався, перебуваючи не в зоні дії давачів SQ3, SQ4, SQ5..

3 ЛАБОРАТОРНА РОБОТА №3
КЕРУВАННЯ ДОЗУВАННЯМ КОМПОНЕНТІВ ФАРБИ

Мета: ознайомитися з пристроєм і функціональними можливостями стенду засобів автоматизації фірми SIEMENS. Оцінити обсяг робіт, які належить виконати в цьому курсі на цьому стенді. Зрозуміти палітру пристроїв автоматизації, ознайомитися з основними функціями контролера S7-200; написати й наладити частину програми, яка відповідає за дозування компонентів фарби по запитам.

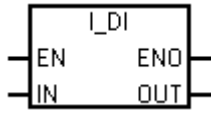
Зрозуміти перспективи використання цього стенда в інших курсах, у дипломному проектуванні, а також у використанні отриманих на ньому навичок для вирішення задач автоматизації й для модернізації існуючих систем управління.

3.1. Теоретичні відомості

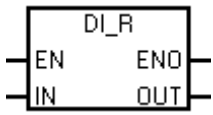
3.1.1 Команди STEP 7 MICRO/WIN

Для реалізації перетворення аналогового сигналу в реальну вагу необхідно використовувати наступні блоки:

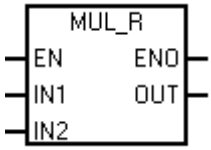
Блок перетворення формату цілих чисел формату (16 біт) у подвійне ціле (32 біт) INT to DINT:



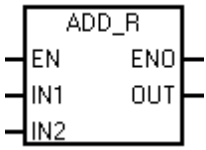
Блок перетворення формату чисел подвійне ціле у реальні DINT to REAL:



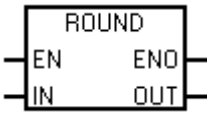
Блок множення двох чисел реального формату REAL:



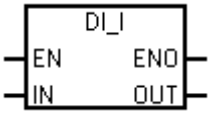
Блок додавання двох чисел реального формату REAL:



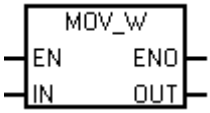
Блок перетворення реального формату чисел в подвійне ціле REAL to DINT:



Блок перетворення формату подвійне ціле число (32 біт) в ціле (16 бит) DINT to INT:



Блок переносу формату інформації WORD (слово):



Позначення входів/виходів блоків.

1. IN1(Input1), IN2(Input2) – відповідно перший та другий вхідні сигнали блоків.
2. OUT(Output) – вихідний сигнал блока

Необхідно врахувати наступне:

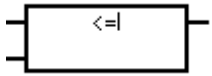
- на вхід EN (Enable), ввімкнення блоку, краще всього подавати SM0.0 (має значення, постійно рівне одиниці 1), тоді наш блок буде мати завжди ввімкнений стан;

- ENO використовується для підтримки наступного блока у ввімкненому стані й сигнал на ньому сигналізує правильність виконання перетворення.

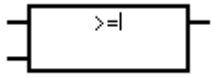
Для того щоб виконати дозування фарби, необхідно порівняти поточну вагу візка й необхідну, для цього необхідно використовувати наступні блоки.

Блоки порівняння двох чисел формату Integer (цілі):

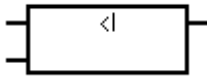
Порівняння «менше або дорівнює»:



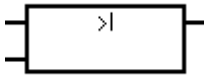
Порівняння «більше або дорівнює»:



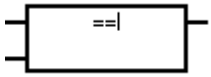
Порівняння «чітко менше»:



Порівняння «чітко більше»:



Порівняння «рівності»:



Таймерні команди SIMATIC. Таймер з затримкою на ввімкнення

Таймер з затримкою ввімкнення з запам'ятовуванням.

Команди «Таймер з затримкою ввімкнення» (TON) та «Таймер з затримкою на ввімкнення з запам'ятовуванням» (TONR) відраховують час, коли вимкнений дозвільний вхід. Номер таймера (Txx) визначає його розподільну здатність, і ця роздільна здатність тепер відобразиться у блоці команди.

Таймерні команди SIMATIC. Таймер з затримкою вимкнення

Таймер з затримкою на вимкнення (TOF) використовується для затримки виходу на фіксований інтервал часу виключення входу. Номер таймера (Txx) визначає його роздільну здатність, й ця роздільна здатність тепер відображається в блоці команди.

Функції таймерів та їх застосування

Таймери виконують різноманітні задачі вимірювання часу:

- таймер з затримкою ввімкнення TON може використовуватись для відліку окремого інтервалу;
- таймер з затримкою ввімкнення з запам'ятовуванням TONR може використовуватись для накопичення декількох відрахованих інтервалів часу;
- таймер з затримкою вимкнення TOF може використовуватись для збільшення інтервалу часу

після вимкнення (або збою), наприклад, для охолодження двигуна після його відключення.

Команди TON та TONR відраховують час, коли ввімкнений дозвільний вхід. Коли поточне значення стає більшим чи дорівнює раніше встановленого часу, біт таймера встановлюється.

- поточне значення таймера TON скидається, коли виключається дозвільний вхід, тоді як поточне значення таймера TONR зберігається, коли цей вхід вимикається;
- ви можете використовувати таймер TONR для накопичення часу, коли цей вхід вмикається й вимикається. Для стирання поточного значення TONR використовується команда Скидання (R);
- таймери TON та TONR продовжують рахунок після досягнення раніше встановленого значення, вони закінчують рахунок при досягненні максимального значення, рівного 32767.

Команди TOF використовуються для затримки вимкнення виходу на фіксований інтервал часу після вимкнення входу:

- коли вмикається дозвільний вхід, негайно вмикається біт таймера, а поточне значення встановлюється в 0;
- коли вхід вимикається, таймер веде рахунок часу, поки відрахований час не досягне раніше встановлене значення;
- коли раніше встановлений час досягнено, тоді біт таймера скидається, а рахунок поточного значення закінчується; однак, якщо вхід вмикається знову, раніше ніж TOF досягає раніше встановленого значення, то біт таймера залишається встановленим;
- щоб таймер TOF почав рахунок часу, до його дозвільного входу повинен бути поданий спадаючий фронт.

Визначення роздільної здатності таймера

Таймери відраховують інтервали часу. Роздільна здатність (або база часу) таймера визначає проміжок часу на один інтервал. Наприклад, TON з роздільною здатністю 10 мс відраховує кількість 10- мілісекундних інтервалів, які пройшли після активації TON: рахунок 50 на 10-мілісекундному таймері представляє 500 мс. Існують таймери SIMATIC з трьома роздільними здатностями: 1мс, 10 мс та 100 мс.

Таблиця 3.1 – Види таймерів

Тип таймера	Роздільна здатність	Максимальне значення	Номер таймера
TON,TOF (із запамя'то-вуванням)	1 мс	32,767 с (0.546 хв.)	T0, T64
	10 мс	327,67 с (5,46 хв.)	T1-T4, T65-T68
	100 мс	3276,7 с (54,6 хв.)	T5-T31, T69-T95
TON,TOF (без запам'ятову-вання)	1 мс	32,767 с (0.546 хв.)	T32,T96
	10 мс	327,67 с (5,46 хв.)	T33-T36, T97-T100
	100 мс	3276,7 с (54,6 хв.)	T37-T63, T101-T255

Формати даних:

- BYTE, WORD, DWORD: ці типи данних визначають вхідний чи вихідний параметр без знака розміром 1,2 або 4 байти відповідно;

- INT, DINT: Ці типи даних визначають вхідний чи вихідний параметр без знака розміром 1,2 або 4 байти відповідно;
- REAL: цей тип даних визначає число з плаваючою крапкою IEEE одноразовою точністю (4 байти);
- STRING: цей тип даних використовується в якості чотирьохбайтового вказника на строку;
- Power Flow [потік сигналу): логічний потік сигналу дозволяється тільки для бітних (булевих) входів.

3.2 Хід виконання лабораторної роботи

Для виконання даної лабораторної роботи необхідно виконати наступні дії.

1. Зробити перекомутацію комутатора стенда, налаштувавши на дозування (за допомогою програми AccessPort: D \ DA_stend \ connect \ _AccessPort).
2. Відкрити модель дозування (на комп'ютері для моделювання D \ DA_stend \ models \ 2_LiquidDozing \ LiquidDozing.mdl).
3. Відкрити STEP 7 Micro / WIN, для програмування конвеєра (на комп'ютері для програмування).
4. Налаштувати привода у програмі STARTER.

3.2.1 Налаштування стенду та комп'ютерів на роботу

Налаштування Access Port та програми STARTER, виконується аналогічно раніше виконаних лабораторних робот.

Відмінною особливістю налаштування програми STARTER для даної лабораторної роботи буде у розділі «Inputs/outputs». Параметром для дискретного входу DI0 необхідно вибрати «Ввімкнути/вимикнути». Для DI1 – фіксована частота (Рис. 3.1).

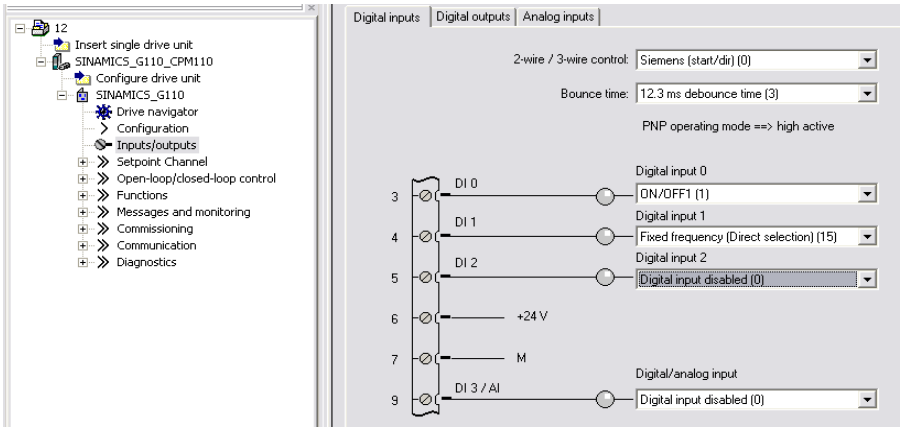


Рисунок 3.1– Налаштування цифрових входів у розділі «Inputs/outputs»

Ці налаштування дадуть можливість реалізувати двохшвидкісний режим дозування фарби. Як виконувалось раніше в лабораторних роботах, необхідно встановити межі фіксованих частот, так мінімальну частоту задати 10 Гц, а максимальну межу 50Гц.

3.2.2 Перетворення аналогового сигналу в одиниці, які використовуються (кілограми)

Однією з задач даної роботи, являється точне визначення ваги набраних компонентів фарби. Зважувальний тензодатчик видає напругу, що пропорційна вазі дозатора на аналоговий вхід AIW0 ПЛК. АЦП цю напругу перетворює в цифрове значення (одиниці АЦП). Для реалізації зручної та зрозумілої роботи, одиниці АЦП потрібно перетворити в одиниці ваги, що використовуються, у даному випадку це кілограми (кг).

Для здійснення перетворення необхідно враховувати те, що вазі порожнього дозатора (0 кг) – G0 вже відповідає деяка кількість одиниць АЦП (JD0).

Залежність показань АЦП від ваги дозатора представлена на рис. 3.2 та підпорядковується лінійному закону:

$$G = k \cdot x + b \qquad (3.1)$$

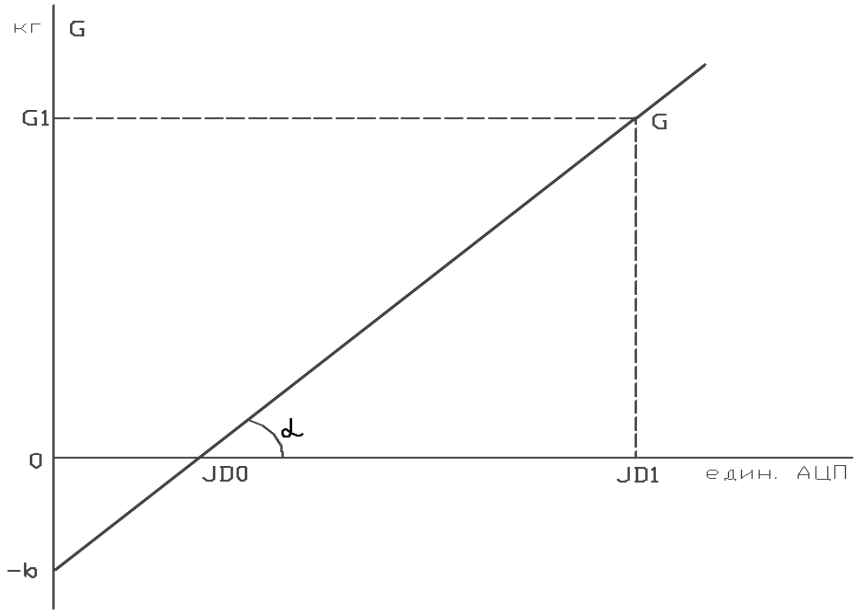


Рисунок 3.2 – Графік залежності кількості одиниць АЦП від ваги дозатора

Виконуємо наступні дії.

1. При вазі дозатора G0, рівній 0 кг для свого стенду визначаємо JD0 (кількість одиниць АЦП). Їх можна спостерігати у вікні Status Chart у строці адреси AIW0, дивися рис. 3.3.

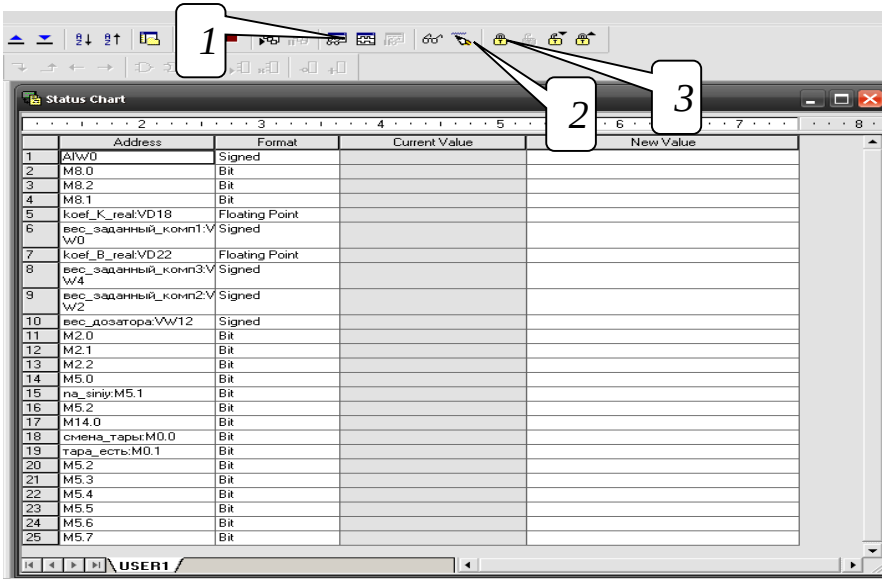


Рисунок 3.3 – Вікно Status Chart

Важливо! Натиснувши кнопку 1 (дивися рис. 3.3) ми отримуємо візуалізацію значень у реальному часі. У стовпчику Current Value спостерігаємо поточне значення (наприклад, задана вага компонента або встановити чи скинути який-небудь запит, то в стовпчику New Value задаємо його нове значення й натискаємо кнопку 2 для запису в пам'ять ПЛК, зміни спостерігаємо в Current Value. Якщо ми хочемо щоб значення, що спостерігається не змінювалось при перезавантаженні, то фіксуємо (режим «Forced») його використовуючи кнопку 3.

2. Наливаємо близьку до максимальної кількості компонент фарби й помічаємо його значення ваги: G1. Беремо значення JD1 з Status Chart як описувалось раніше.

3. Знаходимо коефіцієнт k з лінійного рівняння, який визначає кут нахилу прямої за формулою:

$$k = tg\alpha = \frac{G1 - G0}{JD1 - JD0}, \qquad (3.2)$$

де G1, G0 – максимальна вага, вага порожнього дозатора.

JD1, JD0 – кількість одиниць АЦП при максимальній вазі й вазі порожнього дозатора відповідно.

4. Знаходимо коефіцієнт b, який визначає точку перетину прямої з віссю ваги:

$$b = G0 - k \cdot JD0 = -k \cdot JD0 \qquad (3.3)$$

Отримані коефіцієнти будуть дрібними, тому для більшої точності необхідно працювати з форматом чисел REAL, для цього коефіцієнти k,b повинні бути в форматі REAL.

Щоб ПЛК працював з поточним значенням ваги дозатора, необхідно постійно розраховувати його за формулою 3.1. Для цього значення x (одиниці АЦП) перетворювати постійно в формат REAL, перетворення відбувається за наступним алгоритмом:

$$x(INT) \rightarrow x\_di(DINT) \rightarrow x\_real(REAL)$$

Необхідно використовувати раніше задані локальні змінні, котрі використовують для фіксації проміжних перетворень (рис. 3.4).

Перетворення в STEP 7 Micro/WIN виконується за допомогою блоків, функції яких викладено в теоретичних відомостях лабораторної роботи.

	Symbol	Var Type	Data Type	Comment
LD0	x_di	TEMP	DINT	
LD4	x_real	TEMP	REAL	
LD8	kx_real	TEMP	REAL	
LD12	kxb_real	TEMP	REAL	
LD16	kxb_di	TEMP	DINT	
LW20	kxb_i	TEMP	INT	
		TEMP		

Рисунок 3.4 – Таблиця локальних змінних

3.3 Завдання

В даній лабораторній роботі потрібно написату программу дозування компонент фарби по принципу запит-підтвердження виконання згідно рис.4.5.

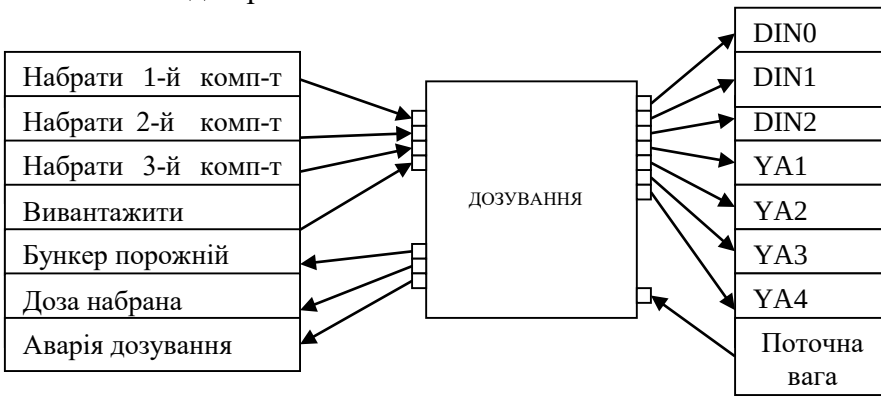


Рисунок 3.5 – Блок-схема програми роботи дозування

Для перевірки роботи програми візок пересуваємо вручну, натисканням відповідних кнопок в моделі LiquidDozing (дивися рис. 3.7) до потрібного положення візка в становлення відповідних запитів згідно часової діаграми процесу дозування, яка наведена на рис. 3.6.

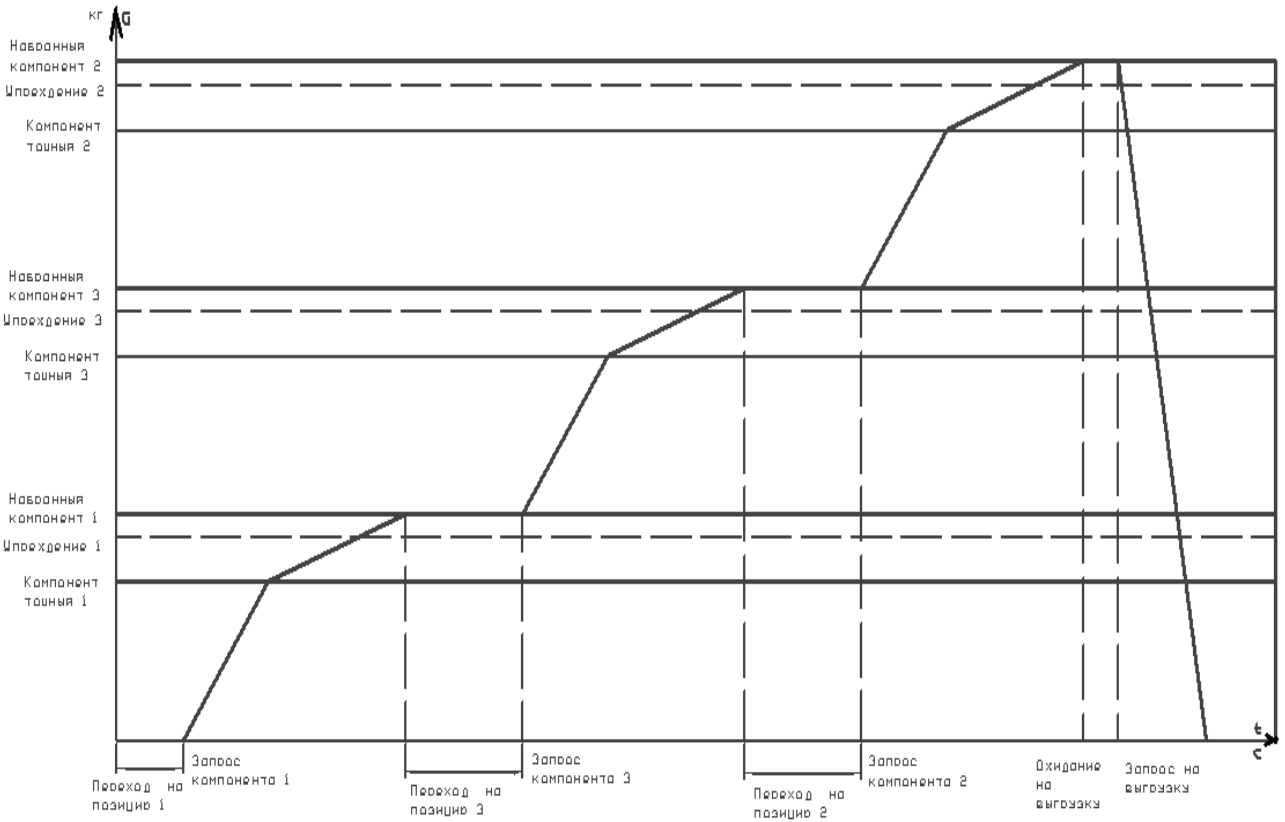


Рисунок 3.6 – Часова діаграма процесу дозування компоненту фарби

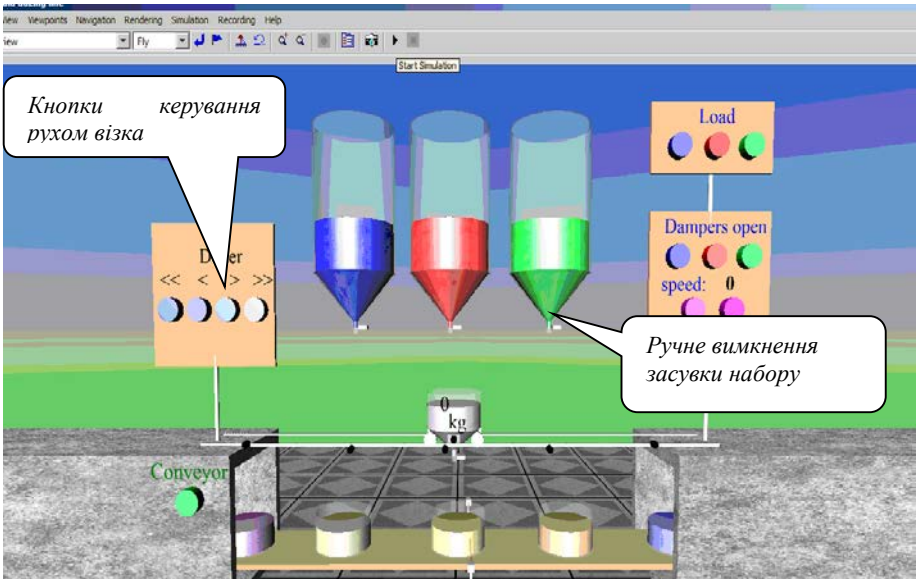


Рисунок 3.7 – Модель дозування компонентів фарби в середовищі MatLab

Послідовність дій виконання завдання.

1. Вручну, як вказувалось раніше, пересуваємо візок в положення 1, потім по встановленню запита на дозування компонента 1 (біт запиту в комірці пам'яті з символічною назвою «набрати_комп1»), повинна відкритися затулка YA1 та почати працювати насос, який приводиться у дію двигуном, що підключений до приводу SINAMIC G110.

2. Вага компонентів фарби, що дозується повинна бути задана на сенсорній панелі TP170micro в стовпчику комірок введення/виведення «ЗАДАНО» (записується в комірку пам'яті з символічною назвою «вес_ заданный_комп1»), а результати набирання налитих компонентів фарби виведення «НАБРАНО» (записується в комірку пам'яті з символічною назвою «вес_набранный_комп1») (Рис. 3.8).

3. Реалізувати дві швидкості дозування фарби: точне та звичайне дозування. Реалізувати двошвидкісне дозування для точного дозування. Виконувати перехід на точне дозування за 10 кг до досягнення заданої ваги, для цього розраховуємо час переходу та записуємо його в комірки пам’яті з символічною назвою «вес_синий_переход_точн».

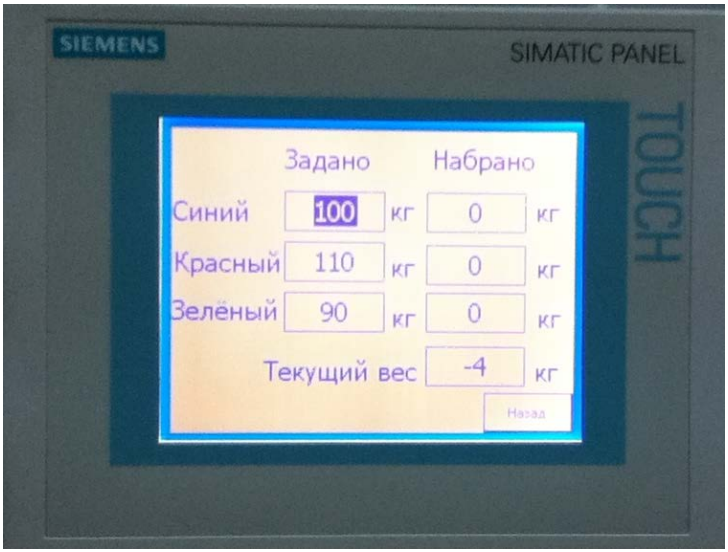


Рисунок 3.8 – Приклад завдання та результатів налитих компонентів фарби на панелі TP170micro

4. Врахувати так іменоване попередження, значення якого визначити експериментальним шляхом. У зв’язку з інерційністю робочих механізмів і тим, що частина фарби все ще знаходиться у повітрі, набір фарби необхідно завершувати (закривати затулку й вимикати насос) раніше на величину попередження, щоб уникнути передозування компонентів фарби. На підставі значення попередження розраховуємо вагу завершення дозування відповідного компонента фарби й записуємо його у

комірки пам’яті з символічною назвою «вес_синий_останов». По закінченню дозування компонента встановити біт виконання «Доза_набрана» й скинути запит на дозування.

5. Вивести вагу набраного компонента фарби в візок на панель TP170micro в відповідне поле стовпця «НАБРАНО» (записується в комірки пам’яті за символічною назвою «вес_набранный_комп»). Для виведення реального значення набору використовуйте затримку фіксації його значення за допомогою таймерних функцій, які описувалися в теоретичному розділі.

6. Вручну пересунути візок у положення 3, а пізніше й у положення 2; реалізувати відповідні запити та повторити попередні пункти для кожного наступного компонента.

7. Коли візок буде знаходитись у положенні 2, біт виконання «Доза_набрана» встановлений і запит на дозування скинутий, необхідно зробити вивантаження компонентів фарби. Для його по запиту з символічною назвою «Дозатор_выгрузить» відкривається засувка візка YA4, яка знаходиться у відкритому стані по досягненні нульової ваги візка. Після цього встановлюють біт виконання «Дозатор_пуст».

УВАГА! Необхідно врахувати, що в лабораторній роботі виконується багатокомпонентне дозування компонентів фари. Тобто, при дозуванні наступних компонентів, маси (вага переходу на точно, вага дозування й набрана вага) повинні враховувати вагу вже набраних компонентів.

3.4. Контрольні питання

1. Назвати основні формати даних та їх особливості.
2. Таймер з затримкою на ввімкнення.
3. Таймер з затримкою на вимкнення.
4. Реалізація перетворення одиниць АЦП в кілограми.
5. Необхідність використання попередження
6. Необхідність використання двошвидкісного дозування компонентів фарби.

ЛІТЕРАТУРА

1. SIEMENS. SIMATIC: Компоненти для комплексної автоматизації / Інформація по продуктах 2010. - Німеччина: 2010г.-167с.
2. SIEMENS. SIMATIC HMI: Human Machine Interface Systems. - Federal Republic of Germany: 2002/2003. - 247p.
3. SIMATIC Программируемый контроллер S7-200. Руководство по эксплуатации - SIEMENS AG 2004- 524с.
4. Мюллер Ю. Регулювання на основі SIMATIC: Практичний посібник з регулювання на основі SIMATIC і SIMATIC PCS7. / Ю. Мюллер - Німеччина: 2002. - 42с.