1.2.2. LD-мова

Мова Крокових діаграм LD (Ladder Diagram) або релейних діаграм, або релейної логіки у графічному вигляді зображає алгоритм розв'язання задач логіко-командного управління. Це одна з мов програмування мікропроцесорних контролерів, які вперше були використані для заміни релейно-контактних схем керування. Тому у вітчизняній техніці ця мова мала саме таку назву –"Мова релейно - контактних схем".

Для опису логічних виразів різного рівня складності ця мова використовує, як базові елементи програмування, графічні елементи "контакти" (contacts) і "котушки" (coils), сполучені відповідно з вхідними та вихідними каналами. Зважаючи на свої обмежені можливості, LD-мова доповнена привнесеними засобами – (таймерами, лічильниками, блоками порівняння, обчислювальними блоками, ПІД-регуляторами та ін.), що дає можливість розв'язувати інші більш складні задачі.

Основою створення технологічної LD-мови є різні варіанти мови релейно-контактних схем фірм виробників ПЛК- контролерів (Allen-Bredley, AEG Schneider Automation, GE-Fanuc, Siemens).

1.2.2.1. Синтаксис LD-мови

Графічні символи LD-мови зображаються всередині програми так саме, як і елементи релейної автоматики в електричній схемі.

Для зображення вхідних контактів використовуються символи:

прямий контакт;

інвертований контакт.

А для зображення вихідних котушок використовуються символи:

пряма котушка;

інвертована котушка;

SET котушка;

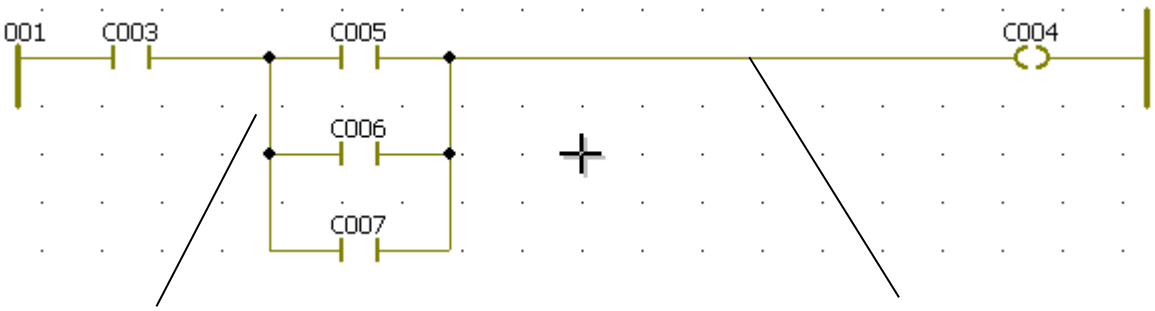
RESET котушка.

Над графічними символами пишеться ім'я змінної, яке фактично є його назвою. Зазвичай, при створені програми графічним символам автоматично присвоюється типове ім'я з відповідним номером, яке може коригуватися користувачем на власний розсуд.

Кількість вхідних контактів у LD- діаграмі може бути скільки завгодно, а вихідна котушка тільки одна. При необхідності паралельно можна вмикати декілька котушок, проте послідовне вмикання їх не допускається.

LD-діаграма обмежена справа і зліва вертикальними лініями, які називаються відповідно лівою шиною живлення і правою шиною живлення.

Графічні символи LD-діаграми сполучені з шинами живлення або іншими символами за допомогою сполучних ліній. Сполучні лінії можуть бути горизонтальними або вертикальними.



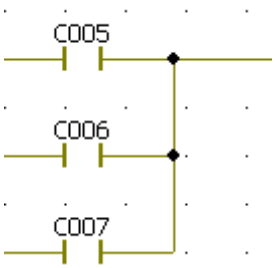
Вертикальні сполучні лінії Горизонтальні сполучні лінії

Кожен сегмент лінії має стан FALSE або TRUE. Сегменти, сполучені разом, мають один і той же логічний стан. Будь-яка горизонтальна лінія, сполучена з лівою шиною живлення, має стан TRUE.

Комбінуючи вертикальні і горизонтальні лінії, можна будувати множинні сполучення. Логічний стан кінців множинного сполучення визначається правилами логіки**.**

Ліве множинне сполучення об'єднує декілька горизонтальних ліній, сполучених з вертикальною лінією зліва і одну горизонтальну лінію, сполучену з її правою стороною. Булевий стан правого кінця є ЛОГІЧНИМ АБО (OR) між всіма лівими кінцями.

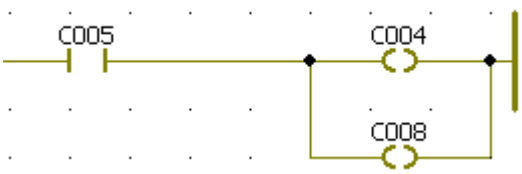
Приклад:



Стан правого кінця:= (C005 OR C006OR C007) *)

Праве множинне сполучення об'єднує одну горизонтальну лінію, сполучену з вертикальною лінією з лівого боку, і декілька горизонтальних ліній, що підходять до її правої сторони. Булевий стан лівого кінця розповсюджується на всі праві кінці.

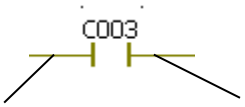
Приклад:



Наведена схема еквівалентна виразу: C004:=C005; C008:=C005.

Множинні сполучення зліва і справа об'єднують декілька горизонтальних ліній, сполучених з вертикальною лінією з лівого боку і декілька горизонтальних ліній, що підходять до її правої сторони. Булевий стан кожного правого кінця є ЛОГІЧНИМ АБО (OR) між всіма лівими кінцями.

Прямий контакт дозволяє виробляти логічні операції між станом лінії і логічної змінної.



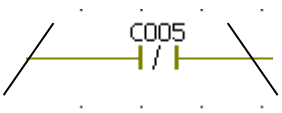
Ліве сполучення Праве сполучення

Стан правого кінця лінії сполучення є логічним ’I’ (AND) між станом лівого кінця і значенням змінної, асоційованої з контактом.

Приклад:

Наведена схема еквівалентна виразу: C004:=C003 AND C005

Інвертований контакт дозволяє виробляти логічні операції між станом лінії і запереченням логічної змінної.



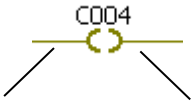
Ліве сполучення Праве сполучення

Стан правого кінця лінії сполучення є логічним ’I’ (AND) між станом лівого кінця і запереченням значення змінної, асоційованої з контактом.

Приклад:

Наведена схема еквівалентна виразу: C004 := NOT (C003) AND NOT (C005);

Пряма котушка визначає логічний вихід стану лінії сполучення**.**

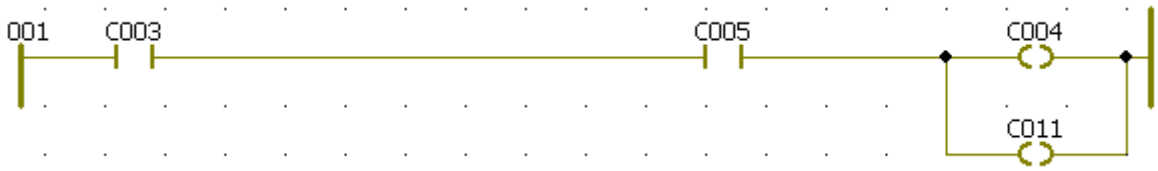


Ліве сполучення Праве сполучення

Змінній котушки призначається стан лівого сполучення. Стан лівого сполучення розповсюджується на праве сполучення. Праве сполучення може бути приєднане до правої вертикальної шини живлення.

Асоційована логічна змінна повинна бути ВИХОДОМ або ВНУТРІШНЬОЮ.

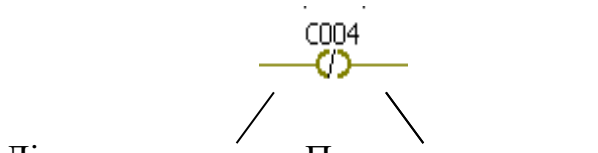
Приклад:



Наведена схема еквівалентна виразу: C004:= C003 AND C005;

C011:= C003 AND C005.

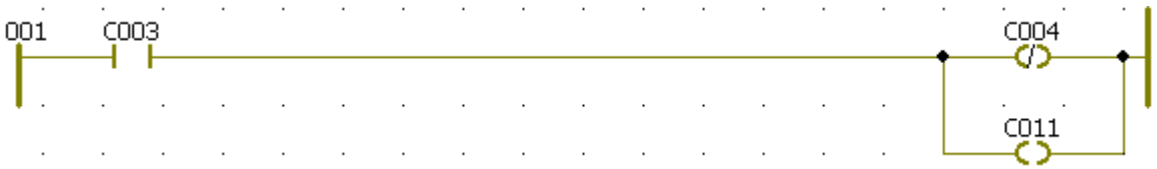
Інвертована котушка визначає логічний вихід, відповідний запереченню стану лінії сполучення**.**



Змінній котушки призначається заперечення стану лівого сполучення. Стан лівого сполучення розповсюджується на праве сполучення. Праве сполучення може бути підключене до правої вертикальної шини живлення.

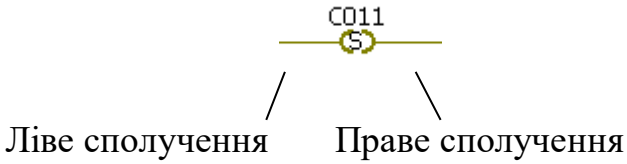
Асоційована логічна змінна повинна бути ВИХОДОМ або ВНУТРІШНЬОЮ.

Приклад:



Наведена схема еквівалентна виразу: C004:= NOT(C003); C011:= C003.

'Set' котушка визначає логічний вихід стану лінії сполучення**.**



Асоційована змінна встановлюється в TRUE (SET TO TRUE), коли стан лівого сполучення стає рівним TRUE. Вихідна змінна утримує цей стан до тих пір, поки вона не буде інвертована котушкою 'RESET'. Стан лівого сполучення розповсюджується на праве сполучення. Праве сполучення може бути підключене до правої вертикальної шини живлення.

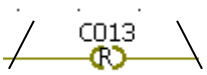
Асоційована логічна змінна повинна бути ВИХОДОМ або ВНУТРІШНЬОЮ.

Приклад:

Наведена схема еквівалентна виразу:

```
IF C003 THEN
    C004 := TRUE;
END_IF;
IF C012 THEN
    C004 := FALSE;
END_IF;
```

"Reset" котушка визначає логічний вихід стану лінії сполучення**.**

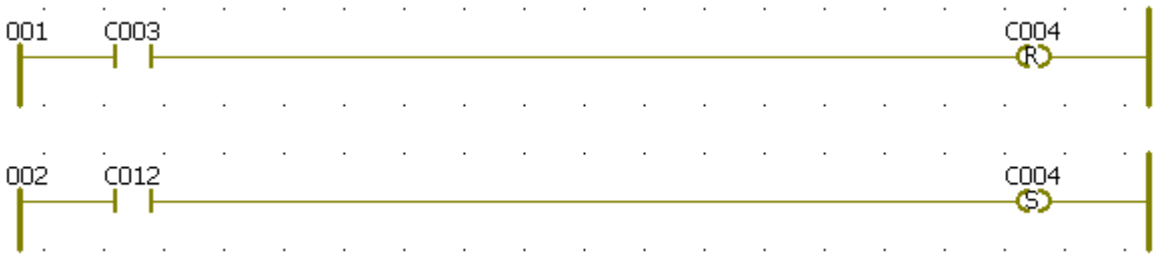


Ліве сполучення Праве сполучення

Асоційована змінна встановлюється в FALSE (RESET TO FALSE), коли стан лівого сполучення стає рівним TRUE. Вихідна змінна утримує цей стан до тих пір, поки вона не буде інвертована котушкою "SET". Стан лівого сполучення розповсюджується на праве сполучення. Праве сполучення може бути підключене до правої вертикальної шини живлення.

Асоційована логічна змінна повинна бути ВИХОДОМ або ВНУТРІШНЬОЮ.

Приклад:



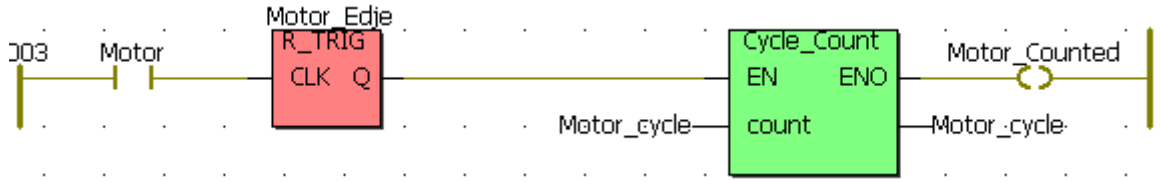
Наведена схема еквівалентна виразу:

```
IF C003 THEN
    C004 := FALSE;
END_IF;
IF C012 THEN
    C004 := TRUE;
END_IF;
```

Окрім розглянутих елементів LD – мови, першою редакцією стандарту IEC 61131-3 ще визначалися контакти та котушки, які керуються фронтами імпульсів: контакт −│P│− і котушка –(P)− переднього фронту , контакт −│N│− і котушка –(N)− заднього фронту. У теперішній час підтримка таких контактів і котушок засобами програмування не є обов'язковою, оскільки аналогічна LD – діаграма легко створюється за допомогою функціональних блоків R_TRIG і F_TRIG.

Таким чином, у LD разом з контактами та котушками можна використовувати функції та функціональні блоки. Оскільки блоки не завжди мають логічні входи і/або виходи, введення блоків в LD- діаграму призводить до додавання в інтерфейс блока нових параметрів – EN та ENO .

У деяких функціях і функціональних блоках перший вхід або перший вихід не є булєвим. Тому при використанні їх в LD- програмах штучно створюються логічний вхід '**EN**' і логічний вихід '**ENO'.** Блок виконується тільки тоді, коли вхід '**EN**' дорівнює TRUE. Вихід '**ENO**' завжди має те саме значення, що і перший вхід блока. У деяких випадках потрібні обидва параметри '**EN**' і '**ENO'.** Нижче наведено приклад використання функціонального блока R_TRIG разом з функцією користувача Cycle_Count (Рахувати цикли), яка має логічний вхід '**EN**' і логічний вихід '**ENO**'.



Послідовність виконання LD–діаграм можна примусово змінити, використовуючи мітки та переходи.

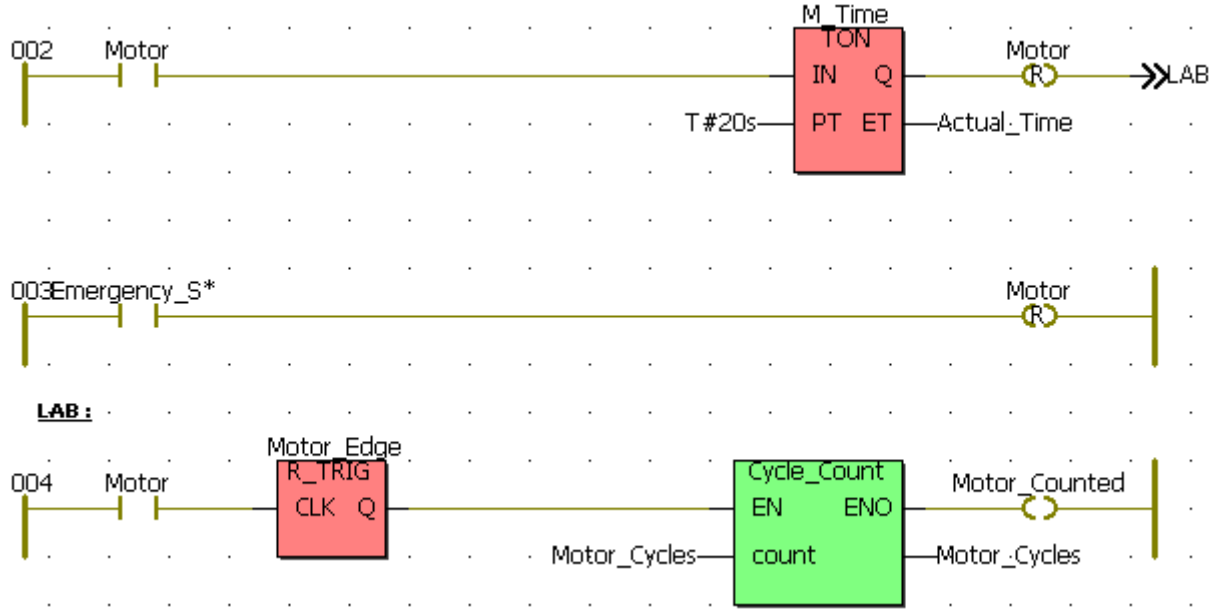
Мітка ставиться тільки на початку LD – діаграми, при цьому для наочності після неї можна ставити дві крапки.

>>LAB - перехід на мітку "LAB";

(LAB:) – мітка на початку діаграми.

LD – діаграма може мати тільки одну мітку та один перехід. Перехід розглядається як вихідне реле і виконується, коли вихідна змінна встановлюється в TRUE . Використовуючи перехід, можна пропустити виконання деяких діаграм.

Приклад:

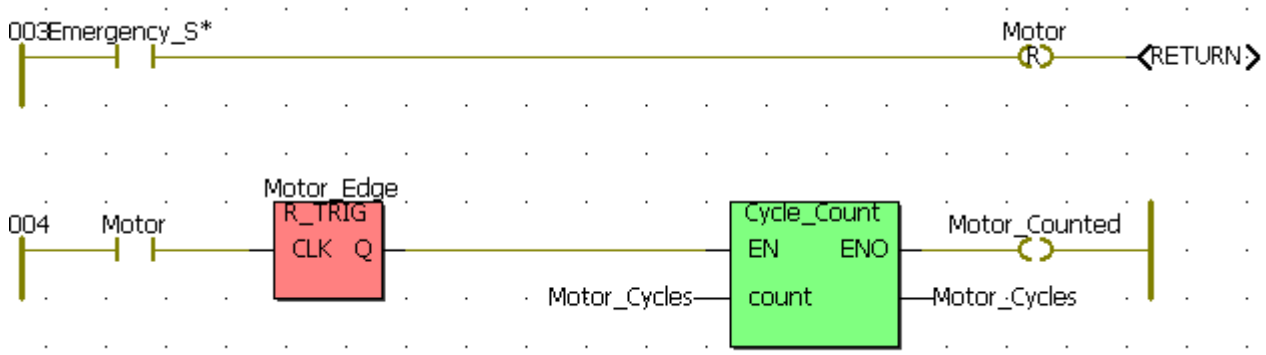


Проте, бажано не займатися керуванням послідовності виконання LD – діаграм, оскільки це суперечить їх аналогії з релейними схемами. Простіше порядок виконання описувати SFC-мовою.

Спеціальний перехід RETURN використовується для умовного завершення програми. Ніяких символів до правого кінця RETURN підключати не можна.

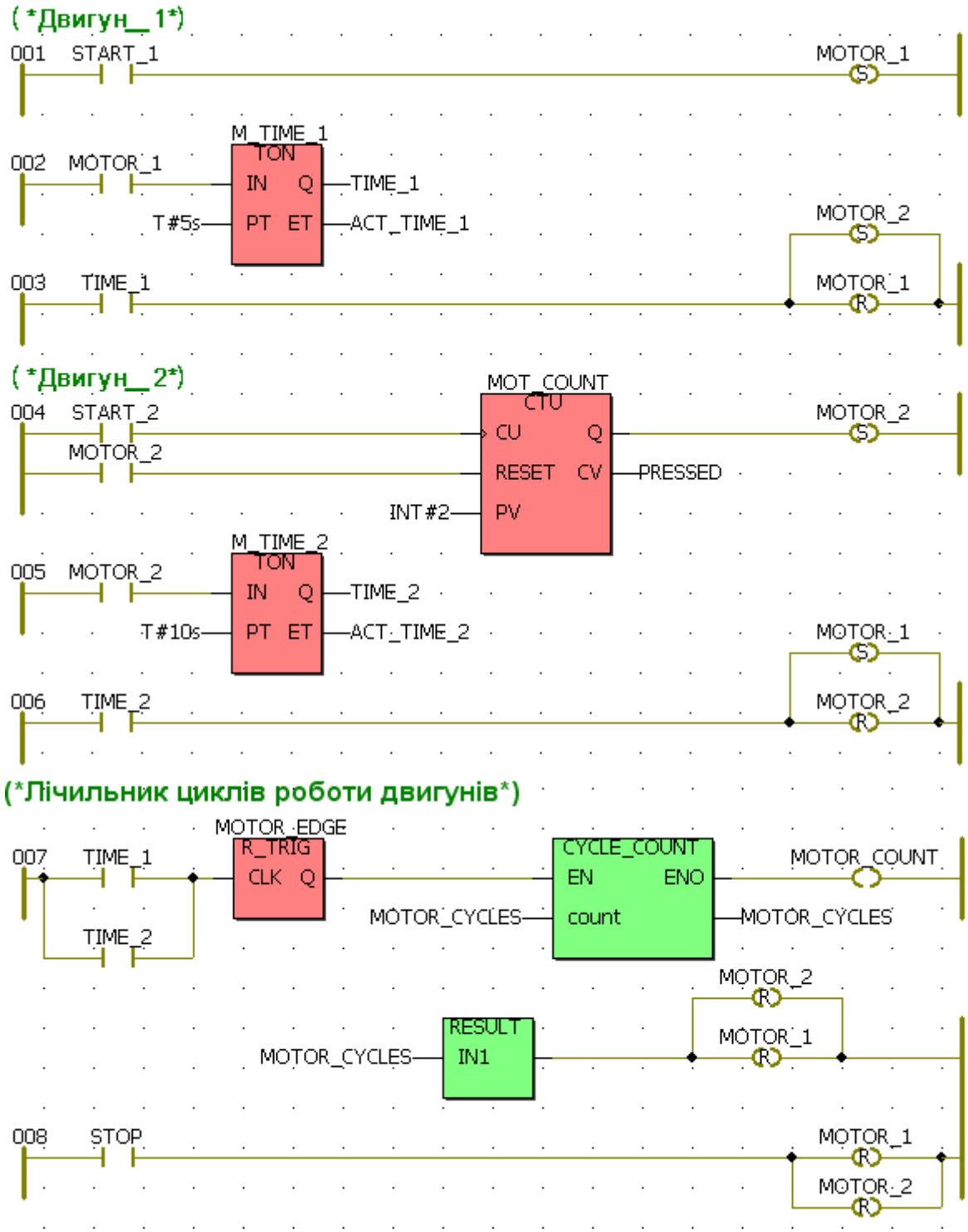
Якщо лівий кінець лінії сполучення має стан **TRUE**, то програма завершується, не виконуючи подальші рядки діаграми[6,7].

Приклад:



1.2.2.2. Приклад програмування LD-мовою

Розробимо програму керування роботою двох двигунів, які після запуску одного з двигунів безперервно по черзі вмикаються і вимикаються. Тривалість роботи двигунів відповідно 5 і 10 секунд. Початковий запуск першого двигуна здійснюється одноразовим натисненням пускової кнопки, а початковий запуск другого двигуна – дворазовим. При цьому двигуни можна у будь-яку мить зупинити, а кожне вмикання двигунів підраховується лічильником. Коли кількість вмикань двигунів досягає десяти, їх робота автоматично зупиняється. Проектний код, що реалізує наведений алгоритм, має вигляд:



Тут:

START_1 і START_2 - кнопки початкового пуску двигунів;

MOTOR_1 і MOTOR_2 – перший і другий двигуни;

M_TIME_1 і M_TIME_2 – екземпляри функціонального блока таймера TON, які відлічують час роботи двигунів;

MOT_COUNT – екземпляр функціонального блока лічильника CTU, який підраховує кількість натисків на кнопку START_2;

MOTOR_EDGE – екземпляр функціонального блока детектора імпульсу переднього фронту R_TRIG.

CYCLE_COUNT(*CYCLE_COUNT:=count+1;)*– функція користувача ST-мовою, яка підраховує активізації двигунів;

RESULT*(RESULT:=IN1=10;)* – функція користувача ST-мовою, яка порівнює поточну кількості вмикань двигунів з уставкою 10;

STOP – кнопка непередбаченої зупинки двигунів.